UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADA EN INGENIERÍA DEL SOFTWARE

DESARROLLO DE UNA APLICACIÓN WEB PARA LA GESTIÓN DE TAREAS DOCENTES EN UN CENTRO EDUCATIVO

DEVELOPMENT OF A WEB APPLICATION TO MANAGE TEACHING TASKS IN AN EDUCATIONAL CENTRE

Realizado por
María José Salas Millán

Tutorizado por
Mónica Pinto Alarcón

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, 2021

Fecha defensa: febrero 2021

Resumen

Actualmente casi todos los centros educativos tienen plataformas web, tanto para profesores como para alumnos, en las que van publicando los horarios, documentos relevantes o todo el material necesario para las clases y que así sea accesible desde cualquier parte y en el momento en el que se necesite. Además, disponen de una plataforma llamada SÉNECA, que pertenece a la Junta de Andalucía, en la cual se encuentra toda la información de un centro educativo la cual se puede exportar en un archivo cuyo formato es CSV, que es fácilmente reutilizable por otras herramientas.

Uno de los objetivos principales de este proyecto es hacer un análisis y diseño completos de todo lo necesario para crear una herramienta que pueda crecer de forma incremental y que sea intuitiva y fácil de utilizar. Esta herramienta estará dedicada a los docentes y permitirá gestionar tanto los datos exportados de SÉNECA como datos nuevos, también debe ejecutar una rutina para aumentar automáticamente los alumnos de curso y dar la posibilidad de crear informes de tutoría.

El otro objetivo principal es, tras el diseño de la base de datos, el análisis y elaboración de casos de uso con sus respectivos diagramas, se implementarán los requisitos más importantes que ha proporcionado el cliente, además de aquellos apartados de gestión necesarios para la funcionalidad básica de la herramienta.

Cada sección de gestión de datos dispondrá de una tabla con el listado de alumnos, profesores o asignaturas, en función del apartado en el que se esté en ese momento, y un buscador en el cual se podrá buscar por cualquier atributo que aparezca en la cabecera de la tabla correspondiente.

Para finalizar se dispondrá de un manual de usuario explicando cómo está dividida la web, como acceder a cada parte, como utilizar el buscador y como añadir, editar, consultar o eliminar datos, así como importar los archivos CSV descargados de SÉNECA, en el caso de que sea necesario su consulta por algún cliente de la aplicación.

Palabras clave: aplicación web, web, colegio, SÉNECA, centro educativo

Abstract

Nowadays, almost every educational center has web pages for students and teachers. These web pages contains schedules, important documents or lessons for students to get prepared for classes, so they can have access to them anytime. Also, they have a platform called SÉNECA which belongs to Junta de Andalucía. There you can find all the information about an educational center which includes, among other things, teachers and students data, groups, subjects. SÉNECA allows to export all the data into a CSV file that can be used in several web pages or tools.

One of the project's main purpose is to make a full analysis and design of what's necessary in order to create an intuitive and easy tool dedicated to teachers so they can manage the data exported from SÉNECA as well as new data. Also, the tool must be able to execute a routine which promotes all the students into a high course automatically and should allow teachers for creating student's reports.

The other main goal of the project is, after the analysis, the database design and the use case development with their graphics, the most important requirements will be implemented as well as all the management sections that are necessary to get the tool's basic functionality. Every section will have a table with the data that belongs to that particular part and a search engine that can be used to search in the table.

To conclude, the tool will have an instructions manual which explains how the web is divided, how to search in the table, to import the CSV file and to add, edit or remove the information.

Keywords: web application, web, school, SÉNECA, educational centre

Índice

1

Introducción

1.1 Motivación

Hoy en día, la mayoría de los centros educativos tienen su página web con toda la información necesaria tanto para padres como para alumnos. Sin embargo, no siempre tienen una aplicación que les ayude en sus tareas diarias. Aunque todos los centros públicos y concertados de Andalucía tienen la aplicación SÉNECA, proporcionada por la Junta de Andalucía, esta aplicación no siempre ofrece todas las funcionalidades que los profesores necesitan cada día para trabajar. Aun así, esta aplicación sí que tiene una característica muy importante que es la posibilidad de exportar los datos de alumnos y profesores para ser reutilizados por otras aplicaciones.

Estos datos importados de la aplicación SÉNECA se pueden importar a una base de datos y combinar con multitud de funciones muy útiles para los profesores, como por ejemplo el ver el listado de alumnos de un curso, buscar los datos de un alumno en particular, mostrar los profesores que da clase a un curso o hacer informes asociados a un determinado alumno en el caso de que sus padres vayan a tutoría.

Por tanto, se ha decidido realizar este proyecto para crear una web que ayude a los profesores de un colegio en particular a realizar sus tareas diarias y además asegurarse de que es intuitiva, fácil de utilizar y modular para que en futuros proyectos se puedan añadir nuevas necesidades, pero que siga manteniendo el diseño y la funcionalidad base de la web.

1.2 Objetivos

El objetivo principal es el diseño y desarrollo de una aplicación de gestión web que cubra las necesidades actuales de un centro educativo de educación infantil y primaria y que además sea intuitiva, robusta y fácil de utilizar para los profesores. Para alcanzar este objetivo principal se considerarán los siguientes objetivos secundarios:

- Identificar las necesidades del centro con respecto a la web. Mediante entrevistas con la dirección del centro se identificará la lista de requisitos, distinguiendo claramente cuáles son requisitos obligatorios y cuáles son requisitos deseables. Dado el tiempo acotado de la realización de un TFG, todos los requisitos, tanto obligatorios como deseables se priorizarán de acuerdo al criterio del centro para el que se desarrolla la aplicación. De esta forma, durante el desarrollo del trabajo se intentarán cubrir todos los requisitos obligatorios por orden de prioridad. También se valorará qué requisitos podrán ser abordables. Aquellos que no puedan ser abordables en este TFG se documentarán apropiadamente para poder continuar el desarrollo de esta aplicación web en posteriores TFGs.

- Hacer una web robusta y funcional con un diseño intuitivo y llamativo. Todo el trabajo de desarrollo que se realice debe ser completo y funcional, debe cubrir las necesidades reales del cliente y debe estar correctamente testado. En este sentido se primará la calidad y robustez del producto desarrollado frente a la cantidad de funcionalidades incluidas en la web.

- Desarrollar la web de forma estructurada y modular para que se puedan añadir futuras modificaciones o módulos. Dado que es previsible que en el tiempo de duración de un TFG no sea posible dotar la web de todas las funcionalidades identificadas durante la captura de requisitos de la misma, un objetivo muy importante del trabajo es realizar un diseño apropiado de la misma, que sea estructurado y modular y que permita con el menor esfuerzo posible integrar nuevas funcionalidades en el futuro.

- Permitir importar datos extraídos de SÉNECA. El objetivo es facilitar al usuario la introducción de datos de nuevos alumnos del centro en cada curso escolar, ya que no tendrá que introducirlos de uno en uno, sino que, al importar cada archivo, se permitirá que los datos que contiene se almacenen directamente en la base de datos y cuando la importación se haya realizado con éxito, estos datos serán visibles para el usuario en la web

1.3 Metodología

Para la realización de este TFG vamos a seguir una metodología de desarrollo ágil, con distintas iteraciones en las que se irá refinando poco a poco la web con nuevas funcionalidades. A continuación, se detallan algunas iteraciones ya identificadas, aunque el número final podrá variar a lo largo de la realización del trabajo.

- **Iteración 1**: Entrevistas con la dirección del colegio para el que se va a desarrollar la página web, para identificar todos los requisitos de la aplicación. Identificación del tipo de requisito (obligatorio/opcional) y del grado de prioridad de cada requisito. Identificación de los datos manejados por la aplicación y su exportación a partir de la plataforma Séneca. También se decidirá la tecnología web a utilizar para el desarrollo de la aplicación, el código de colores y otros aspectos de diseño de la aplicación. Se hará una estimación inicial de los N requisitos que sea posible abordar durante la realización de este TFG. Se intentarán agrupar los requisitos utilizando como criterio el uso de datos comunes de forma que se pueda maximizar el número de funcionalidades de la aplicación. Además, se realizará el diseño inicial de la base de datos en base a los datos extraídos de la plataforma Séneca, se diseñarán los casos de uso necesarios para implementar los requisitos que se abordarán y se introducirán los roles de usuario.

- **Iteración 2 a N:** Cada iteración se encargará de diseñar e implementar cada uno de los requisitos previamente identificados. Se agruparán varios requisitos en una sola iteración si se considera apropiado porque hagan uso del mismo conjunto de datos o similar. Para cada iteración se seguirán todas las fases de desarrollo de un producto software:

    1. Se refinarán los requisitos identificados en la iteración 1 relacionados con esta iteración.
    2. Se diseñará la solución.
    3. Se implementará la solución y se integrará con lo desarrollado en las iteraciones anteriores.
    4. Se pondrá la solución a disposición del cliente para su prueba.
    5. Se corregirán los errores de funcionamiento detectados.

1.4 Estructura de la memoria

Esta memoria se dividirá en seis capítulos distintos, además del anexo y las referencias. A continuación, se hará un breve resumen del contenido que se expone en cada capítulo:

1. **Introducción.** En este primer capítulo, se describirá la motivación y los objetivos que han de cumplirse en la realización de este trabajo, además de la metodología utilizada para hacerlo.
2. **Tecnologías.** En este capítulo se hará una breve introducción de cada tecnología y se expondrán algunas de sus características principales.

3. **Requisitos.** Se definirá detalladamente cada requisito especificado por el cliente, de los cuales se escogerán los más prioritarios que serán implementados en este TFG dejando el resto para futuros TFGs que se basen en este.
4. **Modelado y diseño del sistema.** En este apartado se explicará el modelo tanto del sistema como de la base de datos y las razones por las que se ha diseñado así.
5. **Implementación.** Por cada iteración, exceptuando la primera, se explicará la solución propuesta para cubrir el requisito seleccionado para esa iteración, así como las fases por las que se ha ido pasando dentro de la misma iteración.
6. **Conclusiones.** Para terminar, se hablará de los resultados y conclusiones obtenidas tras la finalización del trabajo. Además, se hablará de las dificultades encontradas durante la realización y las futuras líneas que se pueden seguir a partir de este proyecto.
7. **Anexo.** Se detallará el manual de usuario de la aplicación y una guía de instalación de la herramienta.

2

Tecnologías

En este capítulo se describirán las tecnologías utilizadas para realizar este proyecto, se nombrarán las diferentes características que poseen cada una de ellas.

2.1 PHP

Figura 2.1 Logo de PHP de: https://www.php.net/download-logos.php

Es un lenguaje diseñado para el desarrollo web, el código escrito en este lenguaje es ejecutado por un servidor web y tiene soporte para todo tipo de bases de datos para las cuales existen drivers o se pueden conectar a través del driver ODBC. Permite el manejo de formularios y ficheros, cualquier tipo de operación sobre bases de datos y se puede combinar con JavaScript (apartado 2.2) para que la web tenga más dinamismo.

El hecho de poder insertar código en cualquier página HTML, además de poder combinarlo con JavaScript, poder leer ficheros o crear PDFs a partir de formularios y la facilidad para conectarse con la base de datos y ejecutar cualquier consulta, son los motivos por los que se ha utilizado este lenguaje como base para la creación de esta herramienta en vez de lenguajes como Java, ya que estas características eran necesarias para poder implementar los requisitos que el cliente ha detallado.

2.2 JavaScript

Figura 2.2 Logo de JavaScript de: https://www.javascript.com/

Es uno de los lenguajes más potentes actualmente, no es necesario compilarlo ya que se ejecuta en el lado del cliente, agrega más interactividad y hace las webs más dinámicas. Además, hay librerías programadas en este lenguaje muy útiles para darle una mejor experiencia al usuario con la web. Una de estas librerías es jQuery que se utilizará más adelante en los formularios o para subir archivos a nuestra web.

Además, se utilizará este lenguaje para hacer búsquedas dinámicas en las tablas de datos de cada apartado de gestión, para habilitar botones cuando se haya modificado algún campo, para eliminar registros y actualizar la tabla inmediatamente o para visibilizar o invisibilizar ciertos apartados en función del estado del profesor.

2.3 MySQL

Figura 2.3 Logo de MySQL de: https://www.mysql.com/about/legal/logos.html

Es un sistema gestor de bases de datos relacionales, tiene licencia de código abierto por lo que se pueden modificar el código fuente y utilizar los ficheros libremente lo que lo hace flexible y muy usado actualmente.

Además, es una base de datos muy rápida y posee varias capas de seguridad con encriptación de contraseñas, lo que la hace especialmente buena para este proyecto en concreto ya que el sistema necesitará cargar y acceder a grandes volúmenes de datos y contraseñas que deben estar seguras.

2.4 HTML5

Figura 2.4 Logo de HTML5 de: https://www.w3.org/html/logo/

Es un estándar utilizado para dar estructura y definir el contenido de una web, suele combinarse con CSS (apartado 2.5) para darle un diseño a la web y que sea más atractiva para el cliente.

Este lenguaje se utilizará para crear los formularios de iniciar sesión, registrarse y todos los de añadir, editar y consultar datos de los distintos apartados de gestión de los que dispone la herramienta.

2.5 CSS

Figura 2.5 Logo de CSS de: https://www.w3.org/Style/CSS/Overview.en.html

Es un lenguaje que se usa para dale estilo a los elementos HTML que estructuran la web. Permite crear un archivo con todo el diseño de la web y solo sería necesario hacer una llamada desde el archivo HTML para darle el diseño a la web, con esto el archivo HTML no está tan cargado de código y para realizar cualquier cambio solo habría que hacerlo en el archivo CSS.

Combinado con Bootstrap, se utilizará para diseñar cada una de las páginas de la herramienta. Se creará un estilo común para todas las páginas y en cada etiqueta de HTML en la que sea necesaria darle estilo, se añadirán clases o bien predeterminadas de Bootstrap o bien clases creadas manualmente.

2.6 MagicDraw

Figura 2.6 Logo de MagicDraw de: https://www.nomagic.com/products/magicdraw

Es una herramienta CASE que proporciona ayuda a un ingeniero en la fase de desarrollo y mantenimiento del software y permite realizar modelos y diagramas UML. Además, tiene una interfaz intuitiva, opciones para generar código a partir de un diagrama UML y plugins que se le pueden añadir para aumentar la funcionalidad de esta herramienta.

Se utilizará para realizar los diagramas de secuencia de cada uno de los casos de uso que se detallarán en el apartado 4.2.1 correspondientes con los requisitos que expuso el cliente los cuales se encuentran en el apartado 3.1.

2.7 NetBeans

Figura 2.7 Logo de NetBeans de: https://netbeans.org/

Es un IDE creado sobre todo para Java pero que soporta muchos más lenguajes de programación como PHP o C++. Permite que las aplicaciones se desarrollen a partir de módulos, tiene un debugger integrado para PHP y además se le pueden añadir plugins para añadir funciones que el desarrollador pueda necesitar para su proyecto.

Se podrían haber utilizado otros IDE como eclipse, o editores de texto como Notepad++, Sublime Text o Atom, pero se eligió Netbeans porque es de los mejores IDEs para crear un proyecto y permite ejecutar el proyecto en cuanto se hace un cambio sin necesidad de tener que subir los archivos a un servidor para poder ejecutarlos como sería el caso de los editores de texto.

2.8 XAMPP

Figura 2.8 Logo de XAMPP de: https://www.apachefriends.org/

Es una distribución gratuita de Apache, con licencia de código libre, que contiene MySQL (apartado 2.3), un servidor web Apache y varios intérpretes entre ellos el de PHP (apartado 2.1). Esto permite al desarrollador probar su trabajo, sin que sea necesario tener conexión a internet o un servidor propio al que no pueda acceder siempre que lo necesite.

Hay muchas más opciones para montar un servidor web local como por ejemplo WAMPP o AppServ pero XAMPP, además de ser de las más conocidas, es muy sencillo de instalar, no hay que realizar casi ninguna configuración para que el servidor y MySQL funcionen correctamente y es una distribución conocida ya que se ha usado anteriormente en diversos proyectos en grupo realizados durante el estudio de la titulación.

2.9 BootboxJS

Figura 2.9 Logo de BootboxJS de: http://bootboxjs.com/

Es una librería JavaScript que permite al usuario programar cuadros de diálogo con mensajes de alerta o confirmación, entre otros. Suelen utilizarse para mostrarlos antes de borrar un elemento o tras añadir un conjunto de datos, por ejemplo.

En el proyecto se ha usado para que cuando el usuario pulse el botón eliminar, se muestre una alerta en la pantalla pidiendo al usuario que confirme que desea borrar el registro o, por el contrario, que no desea borrarlo. Hay muchas más alternativas, pero se eligió esta porque se había utilizado anteriormente en otros proyectos.

3

Requisitos

En este capítulo se analizarán y especificarán los requisitos del sistema, divididos en funcionales y no funcionales. Dado que esta aplicación es de desarrollo incremental y se ha realizado todo el análisis, modelado y diseño previo, no todos los requisitos que proporcionó el cliente se han podido incrementar por lo que al final de este capítulo se enumerarán los requisitos que han sido implementados en este TFG y su correspondiente iteración.

Los requisitos que se expondrán más adelante son fruto del consenso entre el cliente y el autor de esta aplicación de gestión web. Tras varias reuniones con el cliente, se han documentado las características que debe tener la aplicación de gestión y las necesidades que debe suplir. Este capítulo de requisitos se correspondería con la primera iteración descrita anteriormente en la metodología.

3.1 Requisitos funcionales

A continuación, se especifican los requisitos funcionales de la aplicación para los que se proporciona una descripción breve, su prioridad y requisitos derivados de este.

- **El sistema permitirá que se importen datos desde SÉNECA**
  - Identificador: RF-01
  - Descripción: para no tener que añadir todos los datos del centro educativo que están en SÉNECA, el sistema tendrá un importador que permitirá que se suban archivos con formato CSV y se añadan directamente a la base de datos.
  - Prioridad: alta

- **El sistema permitirá añadir/modificar/consultar/eliminar nuevos alumnos**
  - Identificador: RF-02

- Descripción: en caso de que llegue un alumno nuevo al centro, el sistema permitirá añadir todos los datos de ese alumno a la base de datos del centro educativo.
  En caso de que algún alumno no pase de curso porque tiene materias pendientes o promocione más de un curso por altas capacidades, el sistema permitirá modificar los datos del alumno.
  En caso de que un alumno se vaya a mitad de curso a otro centro educativo, el sistema permitirá borrar a ese alumno de la base de datos del centro educativo.
- Prioridad: alta

- **El sistema permitirá añadir/modificar/consultar/eliminar nuevos profesores**
  - Identificador: RF-03
  - Descripción: en caso de que llegue un profesor para cubrir una baja temporal o para sustituir a otro profesor, el sistema permitirá añadir todos los datos del profesor nuevo a la base de datos del centro educativo. Además, deberá marcar si este profesor es sustituto o temporal.
    En caso de que un profesor se vaya de baja o vuelva al centro educativo, el sistema permitirá modificar los datos del profesor.
    En caso de que un profesor se vaya a mitad de curso a otro centro educativo, el sistema permitirá borrar a ese alumno de la base de datos del centro educativo.
  - Prioridad: alta

- **El sistema permitirá añadir/eliminar todos los datos de los alumnos de una unidad**
  - Identificador: RF-04
  - Descripción: al iniciar el curso académico, se deberán importar todos los datos de los alumnos que entren al primer curso y se eliminarán los datos de los alumnos que hayan terminado el último curso. En caso de que haya un alumno de último curso repetidor, los datos de este alumno se borrarán también y luego se insertarán manualmente.
  - Prioridad: alta

- **El sistema cambiará de forma automática a todos los alumnos a un curso superior**
  - Identificador: RF-05
  - Descripción: al iniciar el curso académico, todos los alumnos que estaban en el centro el curso pasado deberán avanzar un curso automáticamente. En caso de que haya algún alumno repetidor, los datos de este alumno se modificarán manualmente.
  - Prioridad: alta

- **El sistema permitirá que se registren ausencias de profesores**
  - Identificador: RF-06

- Descripción: si un profesor tiene previsto ausentarse, registrará su ausencia en la web indicando los días que va a ausentarse. Además para cada día de su ausencia le podrá dejar preparado a los compañeros que vayan a sustituirle información sobre las tareas a realizar. Esto puede ser mediante un cuadro de texto donde escriba toda la información necesaria, un cuadro de texto para rellenar información por cada sesión que va a faltar o mostrar el horario del profesor por cada día que va a faltar y que en cada hora se indique la tarea a realizar además de poder subir archivos en caso de que sea necesario.
- Prioridad: alta

- **Los profesores podrán rellenar informes de seguimiento de un alumno**
  - Identificador: RF-07
  - Descripción: si un padre o madre de un alumno solicita una tutoría, el tutor iniciará una solicitud de informe de seguimiento. Esa solicitud debe llegar a todos los profesores que le dan clase al alumno, y así cada uno de ellos podrá cumplimentar un apartado con su valoración. De esta forma, cuando se realice la tutoría el tutor tendrá toda la información recopilada en un mismo informe sin necesidad de tener que buscar personalmente a cada profesor para obtener su valoración sobre el alumno en cuestión.
  - Prioridad: alta

- **El sistema tendrá un buscador**
  - Identificador: RF-08
  - Descripción: en caso de que sea necesario buscar cualquier alumno o profesor, el sistema contendrá un buscador que mostrará todos los resultados que contengan las palabras que se han buscado.
  - Prioridad: alta

- **Los usuarios deberán autenticarse para acceder a la web**
  - Identificador: RF-09
  - Descripción: para poder controlar el acceso a los recursos de la web, lo usuarios tendrán que autenticarse con su correo y contraseña.
  - Prioridad: alta

- **El sistema permitirá mostrar el manual de usuario integrado**
  - Identificador: RF-10
  - Descripción: en caso de que cualquier usuario necesite consultar el manual, el sistema tendrá un apartado que contendrá el manual de usuario de la aplicación web.
  - Prioridad: alta

- **El sistema permitirá añadir más de un tutor a una clase**
  - Identificador: RF-11

- Descripción: en el caso de que el tutor de un grupo esté de baja, el profesor que venga a sustituirlo deberá constar también como tutor del grupo, por lo que el grupo tendría dos tutores.
- Prioridad: moderado

- **El sistema mostrará todos los profesores asociados a una clase**
  - Identificador: RF-12
  - Descripción: para cada grupo, se mostrará el listado de profesores que dan clase a ese grupo.
  - Prioridad: moderado

- **El sistema acotará las actividades que puede realizar cada usuario**
  - Identificador: RF-13
  - Descripción: en función del rol que tenga cada usuario que se autentique en la web, tendrá una serie de acciones que podrá realizar.
  - Prioridad: moderada
  -
- **El tutor de un curso podrá consultar las materias pendientes de cada alumno**
  - Identificador: RF-14
  - Descripción: si un alumno tiene materias pendientes de cursos anteriores, el tutor podrá consultarlas buscando el nombre del alumno y entrando en la página en la que están todos sus datos.
  - Prioridad: moderada

- **El sistema tendrá disponible el horario de cada profesor**
  - Identificador: RF-15
  - Descripción: el profesor deberá tener acceso a su horario para poder consultarlo cuando sea necesario.
  - Prioridad: moderada

- **El sistema permitirá descargar los formularios como archivos de formato PDF**
  - Identificador: RF-16
  - Descripción: en caso de que sea necesario descargar un informe o un formulario, el sistema te permitirá descargarlo en PDF.
  - Prioridad: moderada

- **El sistema tendrá una intranet disponible para los profesores**
  - Identificador: RF-17
  - Descripción: para que los profesores puedan almacenar documentos, entre otras cosas, el sistema dispondrá de una intranet a la que accederán con su correo y contraseña.
  - Prioridad: baja

- **Los profesores podrán acceder a la intranet desde el centro o desde fuera de él.**
  - Identificador: RF-18

- Descripción: en caso de que el profesor esté fuera del centro educativo o en él, deberá poder acceder a la intranet siempre que tenga conexión a internet.
- Prioridad: baja

- **La intranet tendrá un almacén de documentos**
  - Identificador: RF-19
  - Descripción: para que los profesores puedan almacenar archivos y acceder a ellos desde cualquier sitio, la intranet almacenará archivos.
  - Prioridad: baja

- **El sistema tendrá un calendario**
  - Identificador: RF-20
  - Descripción: el sistema tendrá un calendario en el que aparecerán los eventos mensualmente y se podrá pasar a los próximos meses en los que también aparecerán sus eventos, en el caso de que haya.
  - Prioridad: baja

- **El calendario permitirá añadir/consultar/modificar/eliminar eventos**
  - Identificador: RF-21
  - Descripción: al hacer click en un día del calendario, el sistema te llevará a una página en la que aparecerán los eventos de ese día y se podrá añadir, consultar, modificar o eliminar un evento. Si la página está vacía es que no hay eventos ese día.
  - Prioridad: baja

- **El sistema permitirá rellenar actas de ciclo**
  - Identificador: RF-22
  - Descripción: el sistema permitirá rellenar actas de ciclo cada vez que sea necesario, se podrá modificar en caso de ser necesario.
  - Prioridad: baja

- **El sistema permitirá rellenar formularios de incidencias TIC**
  - Identificador: RF-23
  - Descripción: en caso de que haya una incidencia TIC se podrá rellenar un formulario para comunicar esa incidencia y que puedan ir a solucionarla.
  - Prioridad: baja

3.2 Requisitos no funcionales

- **La aplicación web deberá ser sencilla e intuitiva**
  - Identificador: RNF-01
  - Descripción: la web deberá tener una apariencia similar a otras webs disponibles de otros centros educativos con las que los profesores ya están familiarizados.
  - Prioridad: alta

- **El manual de usuario debe estar integrado en la web**
    - Identificador: RNF-02
    - Descripción: la web debe disponer de un manual de usuario en caso de que sea necesario consultarlo.
    - Prioridad: alta

3.3 Requisitos implementados

Dado que esta aplicación se irá completando de forma iterativa en próximos TFGs que continúen a este, no todos los requisitos que ha proporcionado el cliente se han implementado, por lo que a continuación se enumerarán los requisitos que se van a implementar tanto en esta iteración como en las dos próximas.

En la primera iteración, se han realizado los requisitos RF-01, RF-09 y RF-13 que permiten importar los datos que se obtienen de Séneca, que los usuarios puedan registrarse e iniciar sesión y, tras la definición de los roles de usuario, que estos puedan acotar las actividades que realiza cada usuario. Algunas de estas actividades serán desarrolladas en las próximas iteraciones.

En la segunda iteración, se han implementado los requisitos RF-03, RF-06, RF-08 y RF-11, que permiten hacer toda la gestión de datos de los profesores y cambiar su estado a ausente en caso de que sea necesario. Además, se ha programado el buscador insertado en cada uno de los apartados de gestión, los apartados de gestión de unidades y de asignaturas para que se puedan gestionar todos los datos en relación a lo importado de los archivos de Séneca y, en la sección de gestión de unidades, se ha dado la opción de añadir más de un tutor a la unidad.

En la tercera y última iteración, se han implementado los requisitos RF-02, RF-04, RF-05, RF-07, RF-10 y RF-16 que permiten realizar toda la gestión de datos del alumnado, añadir o eliminar los alumnos de una unidad en concreto, promocionar automáticamente a los alumnos al inicio del curso académico, gestionar los informes de tutoría de cada alumno y poder descargarlos en PDF. Además, se añadió el manual de usuario a la web y se realizó el diseño completo haciéndola una web sencilla e intuitiva.

Tras esto, los requisitos que quedan por implementar serían RF-12, RF-14, RF-15, RF-17, RF-18, RF-19, RF-20, RF-21, RF-22 y RF-23, que corresponden a la gestión de eventos del calendario, la muestra de un listado de asignaturas pendientes del alumno y de los profesores que dan clase en una unidad, dar la opción de subir un horario al profesor, la creación de una intranet para subir archivos y que sea accesible tanto desde el centro educativo como desde fuera de él, y la opción de rellenar actas de ciclo e incidencias TIC.

Gran parte de estos requisitos se pueden implementar añadiendo apartados a las secciones de gestión ya implementadas, algunas como el listado de profesores o el listado de materias pendientes del alumno, solo sería necesario añadir un subapartado a la visualización actual de los datos. En cuanto al calendario y sus eventos, puede añadirse a uno de los laterales de la página para que sea visible y añadir otro botón a la página principal para gestionar los eventos.

4

Modelado y diseño de la aplicación web

En este capítulo se procederá a describir el diseño de la base de datos y de la aplicación, así como las decisiones que se han tomado junto con el cliente para realizar el diseño que mejor se adapta a sus necesidades. Este capítulo se correspondería con la primera iteración descrita anteriormente en la metodología, ya que se encargará del diseño de la base de datos inicial y de los casos de uso correspondientes a los requisitos.

4.1 Modelado y diseño de la base de datos

A continuación se mostrará un modelo de la base de datos completa, que se ha diseñado de forma iterativa. Tras el modelo se detallarán cada una de las entidades y su propósito en la aplicación web.

El diseño de las entidades de esta base de datos está condicionado por los ficheros .CSV exportados desde Séneca y por el tipo de tareas que se realizarán en la aplicación sobre estos datos. Si se realizara un diseño desde 0, sin tener en cuenta los datos de SÉNECA y el contenido de los ficheros en los que se exportan, lo más apropiado sería una entidad tutores relacionada con los alumnos, los profesores y las unidades. Al tener que adaptar la aplicación a la importación de los ficheros de Séneca, que era uno de los requisitos principales, se decidió crear las entidades teniendo en cuenta el contenido de estos ficheros que el propio centro educativo ha proporcionado.

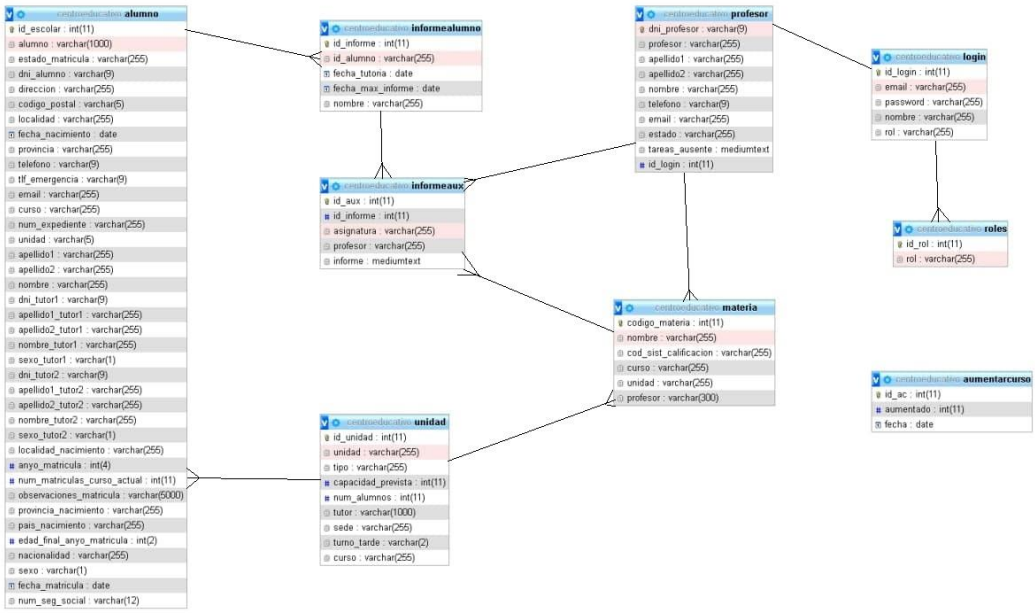

Figura 4.1 Modelo de la base de datos

En la figura 4.1 se encuentran las entidades imprescindibles para que funcione correctamente la herramienta, además de las relaciones que hay entre ellas:

- **Entidad alumno:** es la entidad que almacena todos los datos del alumno, la clave primaria es el i*d escolar* ya que el DNI, el otro atributo candidato a clave primaria, no es obligatorio en menores de 14 años y por lo tanto no todos los alumnos tienen. Esta entidad se relaciona con la entidad *informealumno* de forma que un alumno puede tener varios informes de distintas asignaturas.

- **Entidad informealumno:** es la entidad que almacena una parte los datos de los informes de los alumnos, la clave primaria es *id informe* la cual no es necesario tenerla en cuenta a la hora de hacer una consulta INSERT ya que se incrementa automáticamente. Esta entidad está relacionada con la entidad *materia* y con la entidad *profesor*, ya que cada informe que se realice es sobre una materia que la imparte un profesor. También está relacionada con la entidad *informeaux* y se relaciona mediante el *id_informe*.

- **Entidad informeaux:** es la entidad que almacena todos los datos que inserta el profesor sobre el alumno, es decir, cada apartado del informe que rellene cada profesor estará insertado aquí como un nuevo registro y se relacionará con la entidad *informealumno* mediante *id_informe* que contiene la id del informe global al que pertenecen esos apartados.

- **Entidad profesor:** es la entidad que almacena todos los datos de los docentes, la clave primaria es *dni profesor* ya que es un atributo único para cada persona que lo posea. Esta entidad se relaciona con *informealumno* ya que cada profesor realiza muchos informes y con *materia* ya que un profesor puede impartir muchas materias.

- **Entidad unidad:** una unidad es una división del curso en el que se agrupan los alumnos, es decir, por cada curso hay dos o tres unidades que pueden ser A, B o C. Esta entidad que almacena todos los datos de las unidades de cada curso, la clave primaria es *id unidad* y se autoincrementa automáticamente por lo que no es necesario añadirla en la consulta INSERT. Esta entidad se relaciona con la entidad *materia* ya que en una unidad se imparten muchas materias.

- **Entidad materia:** es la entidad que almacena todos los datos de las asignaturas la clave primaria es *codigo materia*, la clave se autoincrementa automáticamente por lo que no es necesario añadirla en la consulta INSERT. Esta entidad está relacionada con *informealumno*, *profesor* y *unidad*.

- **Entidad login:** es la entidad que almacena los datos necesarios para iniciar sesión y poder utilizar la aplicación, la clave primaria es *id unidad* y se autoincrementa automáticamente por lo que no es necesario añadirla en la consulta INSERT. Además se relaciona con la entidad *roles* en la que se encuentran los distintos roles de usuario que puede tener la persona que ha iniciado sesión.

- **Entidad roles:** es la entidad que contiene los distintos tipos de rol que el usuario que accede a la aplicación puede tener. Esta entidad está conectada a la entidad *login* mediante el atributo rol. Esto se explicará más adelante en el apartado 5.1.7.

- **Entidad aumentacurso:** es la entidad que almacena los datos necesarios para saber si se han aumentado de curso los alumnos al comienzo de cada curso académico.

4.2 Modelado y diseño de la web

4.2.1 Casos de Uso

A continuación, se detallarán los casos de uso más relevantes del sistema. Para cada caso de uso se proporcionarán las precondiciones y postcondiciones que deben cumplir, se detallará el escenario principal y alternativo, en el caso de que exista, y se adjuntará el diagrama de secuencia correspondiente. Para facilitar la continuación de este TFG, se ha decidido llevar a cabo de forma completa tanto la fase de captura de requisitos como la de modelado de los casos de uso. De esta forma toda la información que el cliente ha transmitido durante nuestras reuniones con él se han representado de la forma más precisa posible. De esta forma facilitamos la continuación futura de este trabajo.

4.2.1.1 Importar archivos de Séneca

- **Precondición:** El archivo debe tener extensión CSV.
- **Postcondición:** Los datos que contenía el archivo se han añadido a su correspondiente tabla en la base de datos.
- **Escenario principal:**

1. El usuario hace click en el botón Seleccionar archivo del apartado de gestión en el que se encuentre.
2. El sistema muestra el explorador de archivos del dispositivo.
3. El usuario selecciona el archivo que desea importar y hace click en abrir y en Importar.
4. El sistema muestra un mensaje de que se ha importado correctamente.

- **Escenario alternativo:**
    1. El sistema muestra un mensaje de que el fichero no tiene el formato adecuado.
    2. El sistema vuelve al paso 1 del escenario principal.
- **Diagrama de secuencia:**

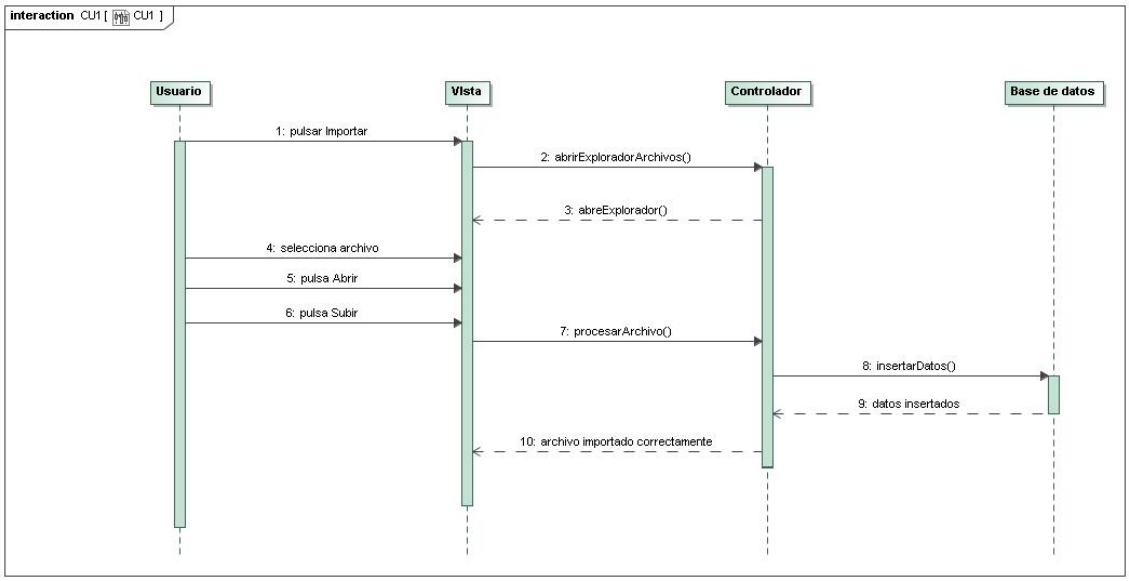

Figura 4.2.1.1 Caso de uso 1

4.2.1.2 Añadir un nuevo alumno

- **Precondición:** El usuario se encuentra en la página Gestión del Alumnado. El alumno no se ha añadido previamente.
- **Postcondición:** El alumno se ha añadido a la base de datos.
- **Escenario principal:**
    1. El usuario hace click en el botón Añadir.
    2. El sistema muestra el formulario con los campos que hay que rellenar.
    3. El usuario rellena los campos y presiona enviar.
    4. El sistema muestra un mensaje de que se ha añadido correctamente y vuelve a la página principal de Gestión del Alumnado.

- **Diagrama de secuencia:**

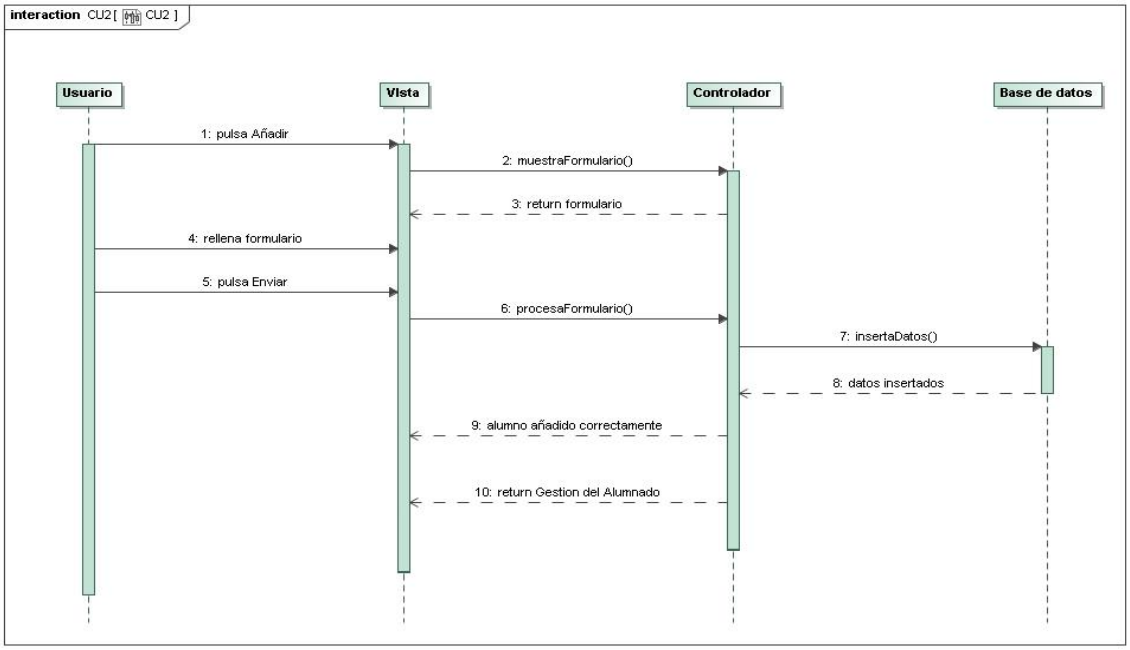

Figura 4.2.1.2 Caso de uso 2

4.2.1.3 Modificar datos del alumno

- **Precondición:** El usuario se encuentra en la página Gestión del Alumnado. Los datos del alumno que desean modificar se ha añadido previamente.
- **Postcondición:** Los datos del alumno han sido modificados en la base de datos.
- **Escenario principal:**
    1. El usuario busca el alumno cuyos datos quiere editar y hace click en el botón Modificar.
    2. El sistema muestra el formulario con todos los campos del formulario rellenos.
    3. El usuario modifica los campos que desee y presiona editar.
    4. El sistema muestra un mensaje de que se ha modificado correctamente y vuelve a la página principal de Gestión del Alumnado.

- **Diagrama de secuencia:**

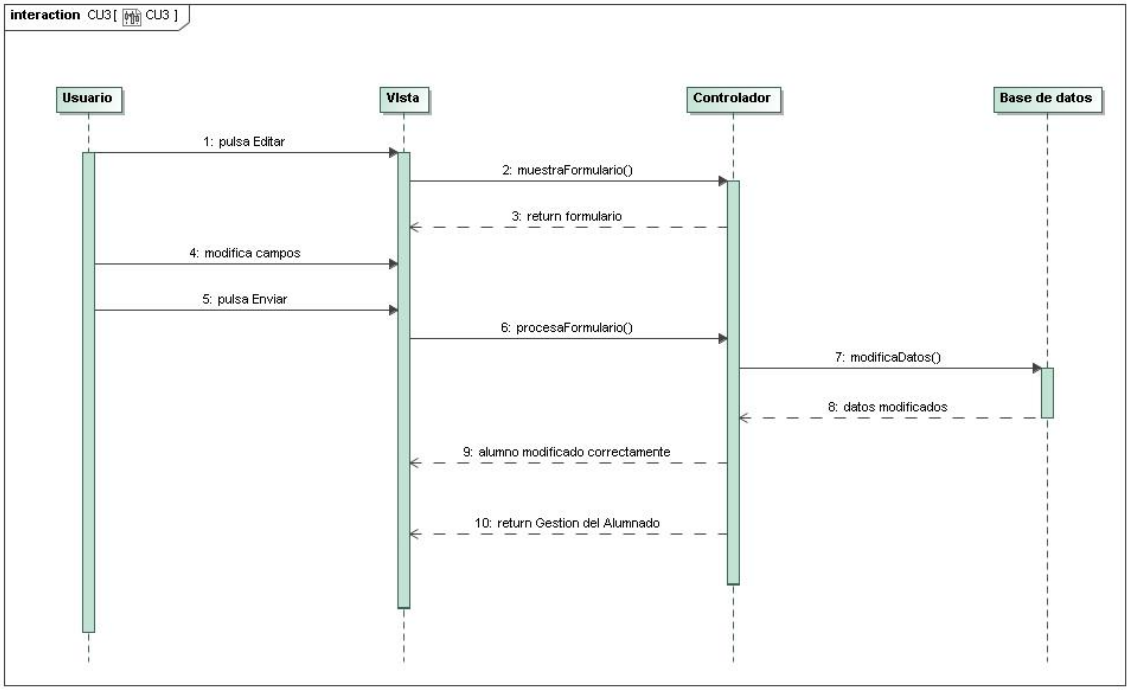

Figura 4.2.1.3 Caso de uso 3

4.2.1.4 Consultar datos del alumno

- **Precondición:** El usuario se encuentra en la página Gestión del Alumnado. El alumno cuyos datos se desean consultar ha sido añadido previamente.
- **Postcondición:** Los datos del alumno se muestran correctamente.
- **Escenario principal:**
    1. El usuario busca el alumno cuyos datos quiere consultar y hace click en el botón consultar.
    2. El sistema muestra los datos del alumno en cuestión.
- **Diagrama de secuencia:**

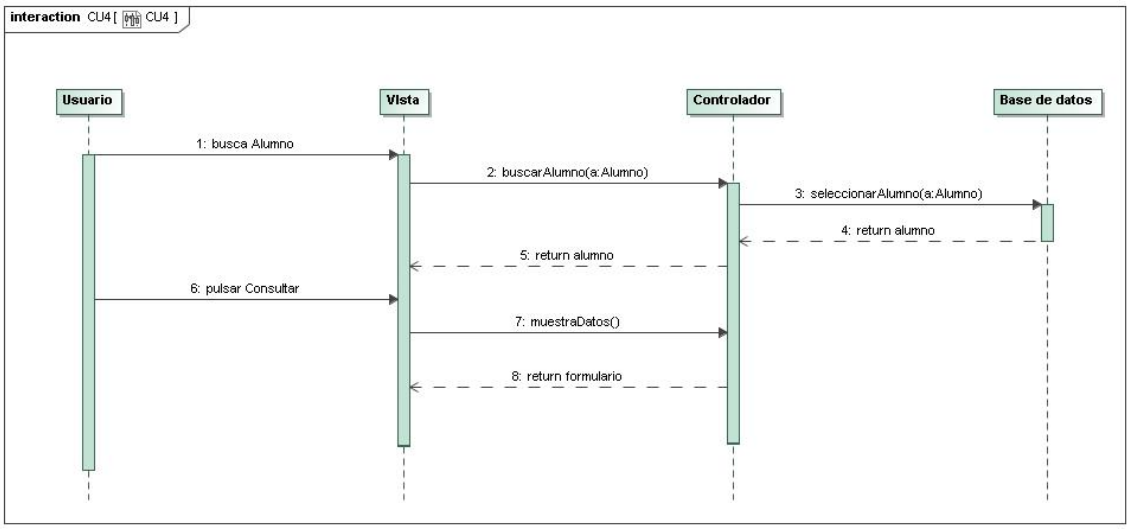

Figura 4.2.1.4 Caso de uso 4

4.2.1.5 Eliminar un alumno

- **Precondición:** El usuario se encuentra en la página Gestión del Alumnado. El alumno que se desea eliminar ha sido añadido previamente.
- **Postcondición:** El alumno se ha eliminado de la base de datos.
- **Escenario principal:**
    1. El usuario busca el alumno que quiere eliminar y hace click en el botón eliminar.
    2. El sistema muestra un subrayado rojo en la línea a eliminar, la elimina y recarga la página principal de Gestión del Alumnado.
- **Diagrama de secuencia:**

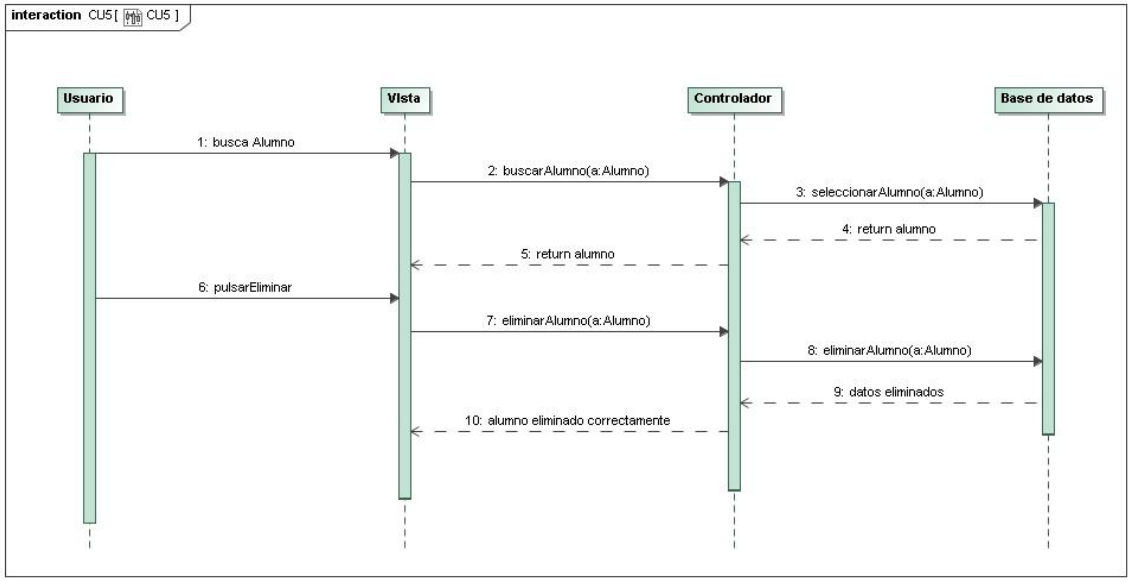

Figura 4.2.1.5 Caso de uso 5

4.2.1.6 Añadir un nuevo profesor

- **Precondición:** El usuario se encuentra en la página Gestión del Profesorado. El profesor no se ha añadido previamente.
- **Postcondición:** El profesor se ha añadido a la base de datos.
- **Escenario principal:**
    1. El usuario hace click en el botón Añadir.
    2. El sistema muestra el formulario con los campos que hay que rellenar.
    3. El usuario rellena los campos y presiona enviar.
    4. El sistema muestra un mensaje de que se ha añadido correctamente y vuelve a la página principal de Gestión del Profesorado.

- **Diagrama de secuencia:**

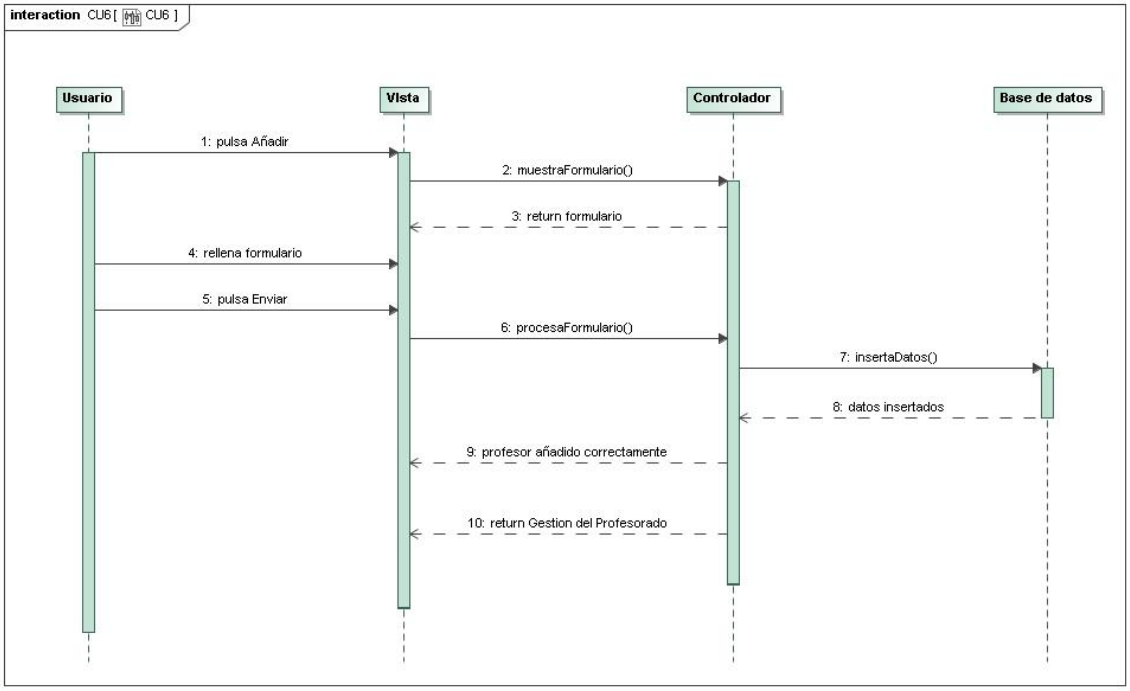

Figura 4.2.1.6 Caso de uso 6

4.2.1.7 Modificar datos del profesor

- **Precondición:** El usuario se encuentra en la página Gestión del Profesorado. Los datos del profesor que desean modificar se ha añadido previamente.
- **Postcondición:** Los datos del profesor han sido modificados en la base de datos.
- **Escenario principal:**
    1. El usuario busca el profesor cuyos datos quiere editar y hace click en el botón Editar.
    2. El sistema muestra el formulario con todos los campos del formulario rellenos.
    3. El usuario modifica los campos que desee y presiona enviar.
    4. El sistema muestra un mensaje de que se ha modificado correctamente y vuelve a la página principal de Gestión del Profesorado.

- **Diagrama de secuencia:**

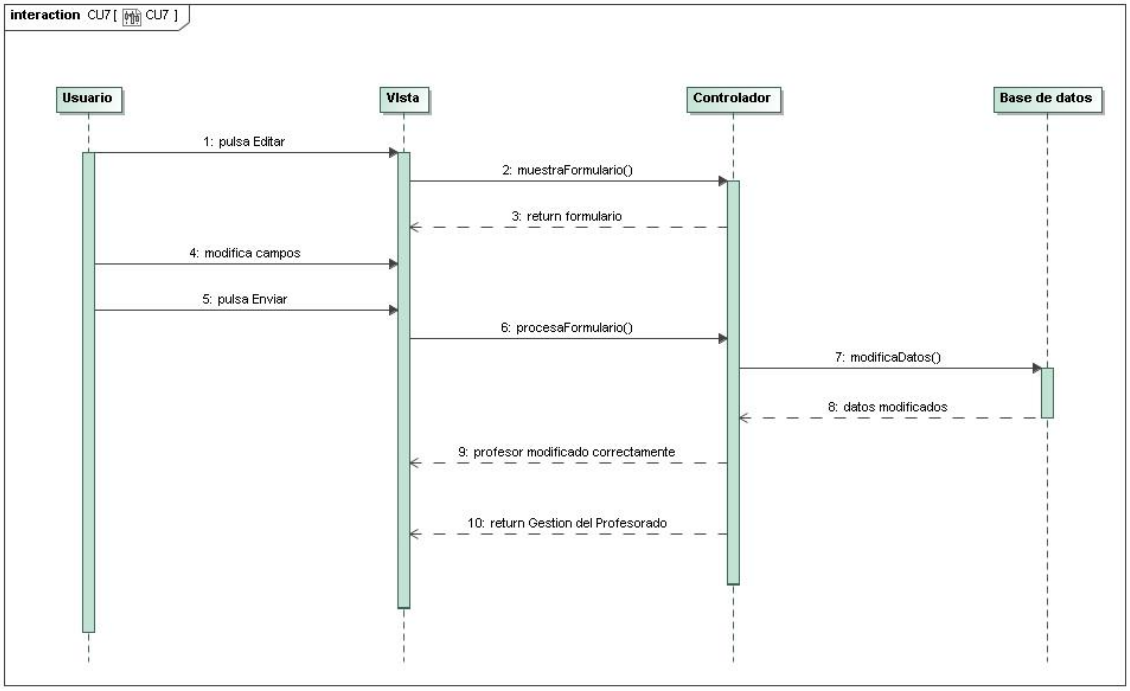

Figura 4.2.1.7 Caso de uso 7

4.2.1.8 Consultar datos del profesor

- **Precondición:** El usuario se encuentra en la página Gestión del Profesorado. El profesor cuyos datos se desean consultar ha sido añadido previamente.
- **Postcondición:** Los datos del profesor se muestran correctamente.
- **Escenario principal:**
    1. El usuario busca el profesor cuyos datos quiere consultar y hace click en el botón consultar.
    2. El sistema muestra los datos del alumno en cuestión.
- **Diagrama de secuencia:**

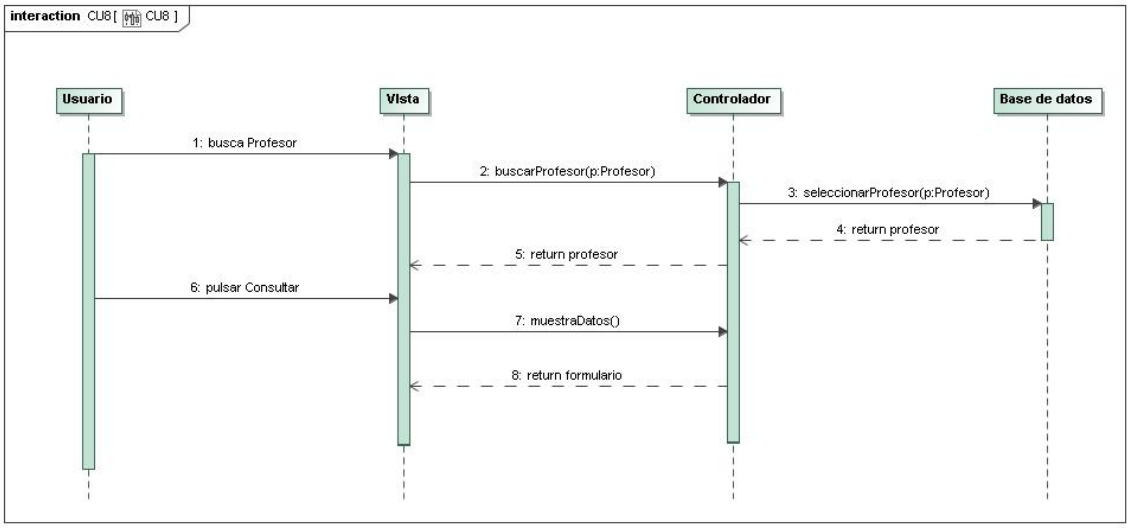

Figura 4.2.1.8 Caso de uso 8

4.2.1.9 Eliminar un profesor

- **Precondición:** El usuario se encuentra en la página Gestión del Profesorado. El profesor que se desea eliminar ha sido añadido previamente.
- **Postcondición:** El profesor se ha eliminado de la base de datos.
- **Escenario principal:**
    1. El usuario busca el profesor que quiere eliminar y hace click en el botón eliminar.
    2. El sistema muestra un subrayado rojo en la línea a eliminar, la elimina y recarga la página principal de Gestión del Profesorado.
- **Diagrama de secuencia:**

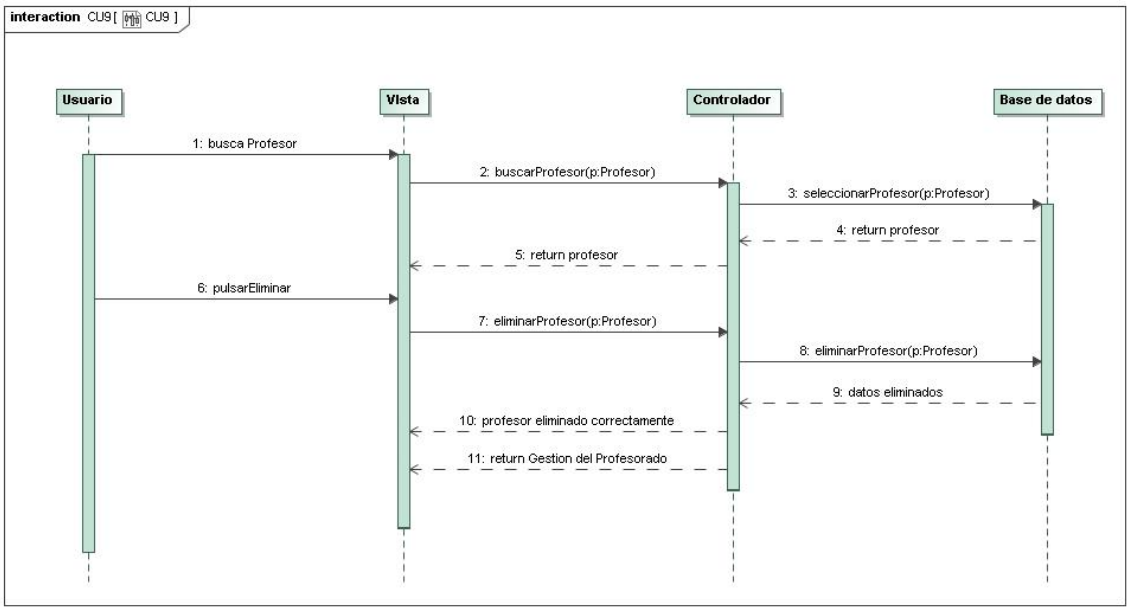

Figura 4.2.1.9 Caso de uso 9

4.2.1.10 Añadir todos los alumnos de una unidad

- **Precondición:** El usuario se encuentra en la página Gestión del Alumnado. Los alumnos de esa unidad en concreto no están en la base de datos.
- **Postcondición:** Los alumnos añadidos están en la base de datos.
- **Escenario principal:**
    1. El usuario hace click en el botón Seleccionar archivo.
    2. El sistema muestra el explorador de archivos del dispositivo.
    3. El usuario selecciona el archivo que desea importar y hace click en aceptar y en Importar.
    4. El sistema muestra un mensaje de que se ha importado correctamente y vuelve a la página principal de Gestión del Alumnado.
- **Escenario alternativo:**
    1. El sistema muestra un mensaje de que la unidad ya tiene alumnos o que el alumno que se va a introducir ya está en la base de datos.
    2. El sistema para la importación y vuelve al paso 1 del escenario principal.
- **Diagrama de secuencia:**

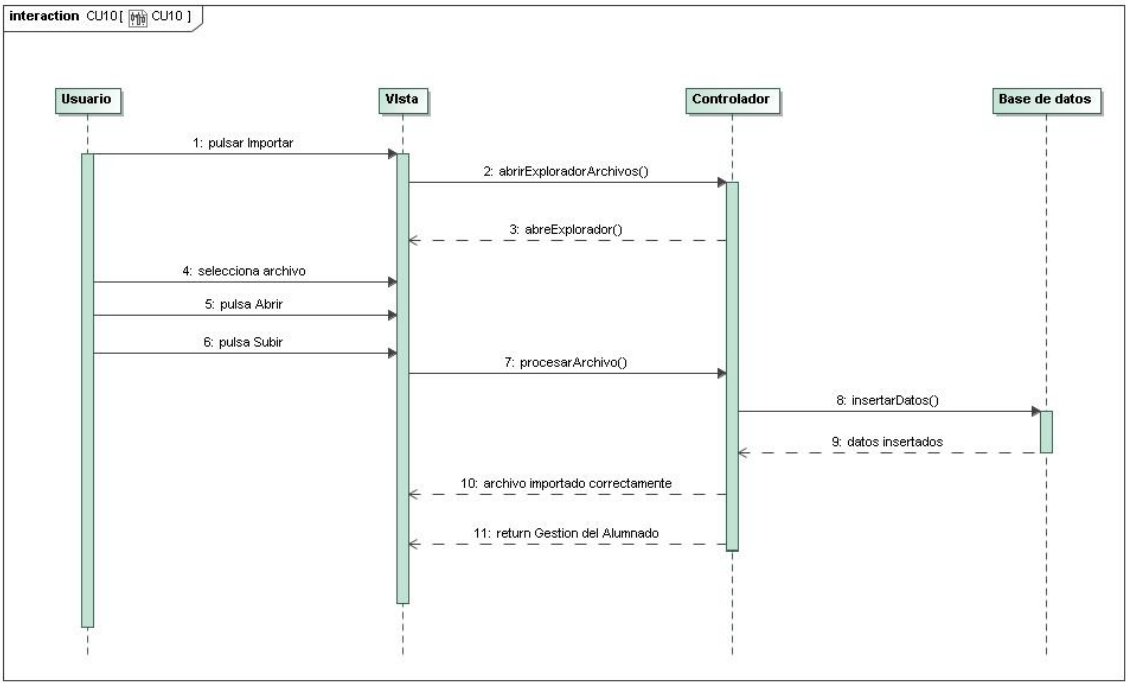

Figura 4.2.1.10 Caso de uso 10

4.2.1.11 Eliminar todos los alumnos de una unidad

- **Precondición:** El usuario se encuentra en la página Gestión del Alumnado. Los alumnos añadidos están en la base de datos.
- **Postcondición:** Los alumnos se han eliminado de la base de datos.
- **Escenario principal:**
    1. El usuario rellenará un campo para introducir la unidad cuyos alumnos quiere eliminar que se encuentra debajo de Eliminar alumnos de una unidad.
    2. El usuario introducirá el nombre de la unidad y pulsará el botón Eliminar.
    3. El sistema mostrará un mensaje de que se ha eliminado correctamente y volverá a la página principal de Gestión del Alumnado.
- **Escenario alternativo:**
    1. El sistema no encuentra la unidad cuyos alumnos se quiere eliminar o la unidad no tiene alumnos.
    2. El sistema vuelve al paso 1 del escenario principal.
- **Diagrama de secuencia:**

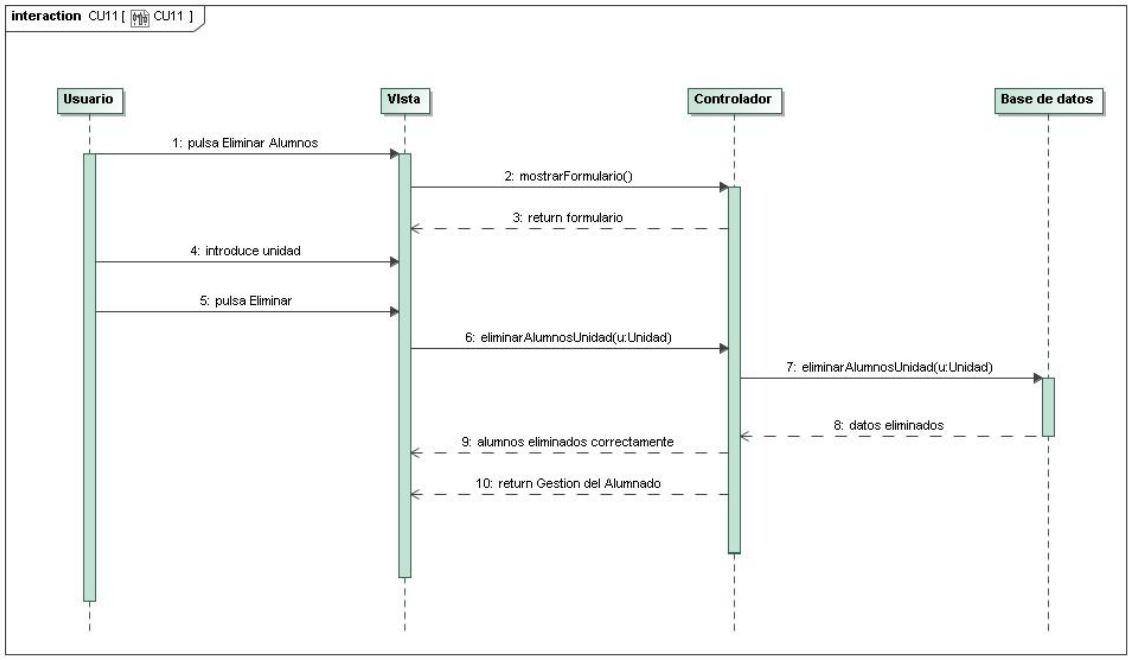

Figura 4.2.1.11 Caso de uso 11

4.2.1.12 Añadir más de un tutor a una clase/unidad

- **Precondición:** El usuario se encuentra en la página Gestión de Unidades. Que la unidad ya tenga asignado un tutor.
- **Postcondición:** Que la clase tenga asignado más de un tutor.
- **Escenario principal:**
    1. El usuario buscará en el listado la unidad a la que le quiere añadir otro tutor y hará click en el botón Modificar.
    2. El sistema mostrará un formulario con los datos de la unidad seleccionada.
    3. El usuario se irá al apartado tutores en el que escribirá el nombre del nuevo tutor junto con su fecha.
    4. El usuario hará click en el botón Enviar.
    5. El sistema mostrará un mensaje de que se ha añadido correctamente y vuelve a la página principal de Gestión de Unidades.

- **Diagrama de secuencia:**

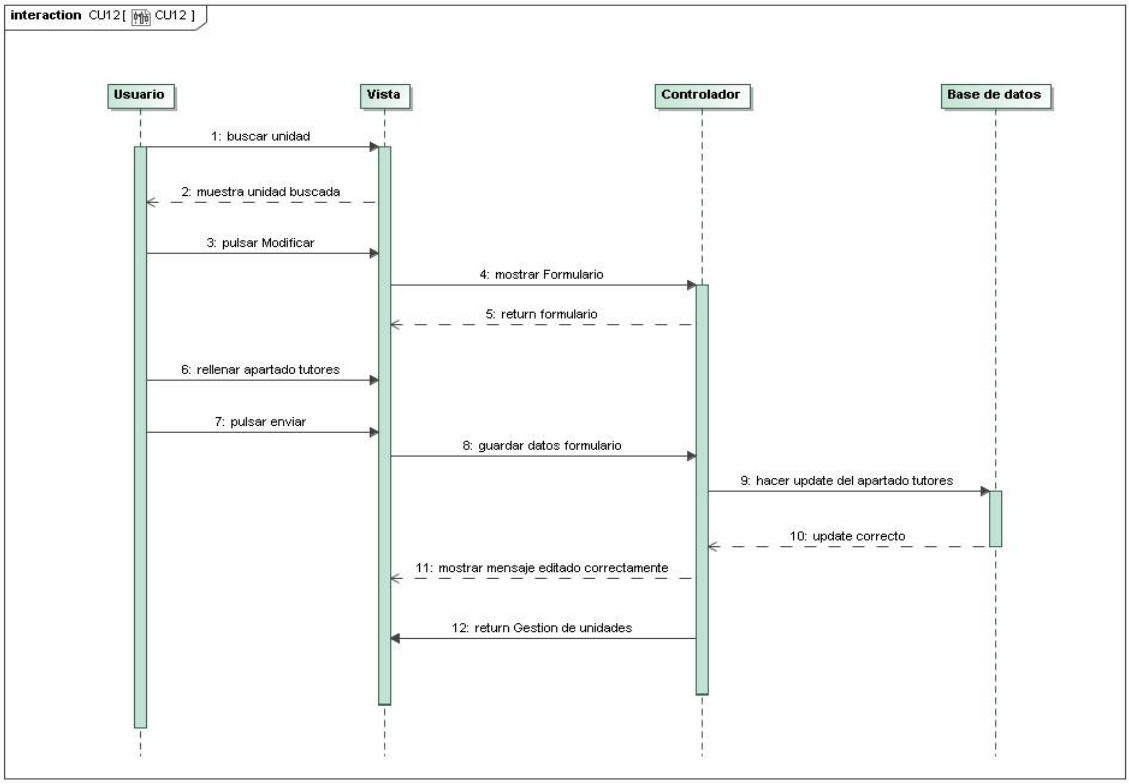

Figura 4.2.1.12 Caso de uso 12

4.2.1.13 Buscador del sistema

- **Postcondición:** Que el sistema haya devuelto una lista con los términos que coinciden con la búsqueda realizada.
- **Escenario principal:**
    1. El usuario introducirá cualquier cadena de caracteres que pueda coincidir con algún dato de la tabla.
    2. El sistema mostrará una lista actualizada con todos los profesores o alumnos que coincidan con los datos introducidos en el buscador, conforme se va escribiendo.
- **Escenario alternativo:**
    1. La búsqueda devuelve una lista vacía porque no coincide con ningún nombre o asignatura.
- **Diagrama de secuencia:**

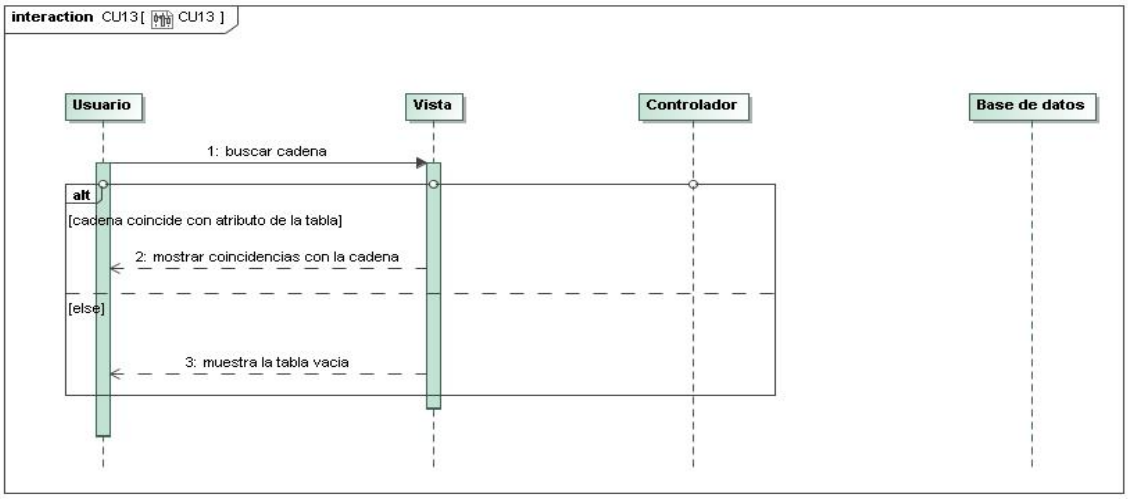

Figura 4.2.1.13 Caso de uso 13

4.2.1.14 Rellenar informe de seguimiento

- **Precondición:** El usuario se encuentra en la página de Gestión del Alumnado y ha buscado el alumno cuyo informe quiere realizar.
- **Postcondición:** Que el sistema haya almacenado los datos rellenos en el informe.
- **Escenario principal:**
    1. El usuario pulsará en el botón Informes.
    2. El sistema mostrará un listado con las informes de seguimiento y un botón Nuevo informe para crear uno nuevo.
    3. El usuario pulsará el botón Nuevo informe.
    4. El sistema le mostrará un formulario con los campos necesarios para rellenar.
    5. El usuario rellenará los campos y pulsará el botón enviar.
    6. El sistema mostrará un mensaje de que se ha creado correctamente el informe y volverá a la página principal de Gestión del Alumnado.
- **Diagrama de secuencia:**

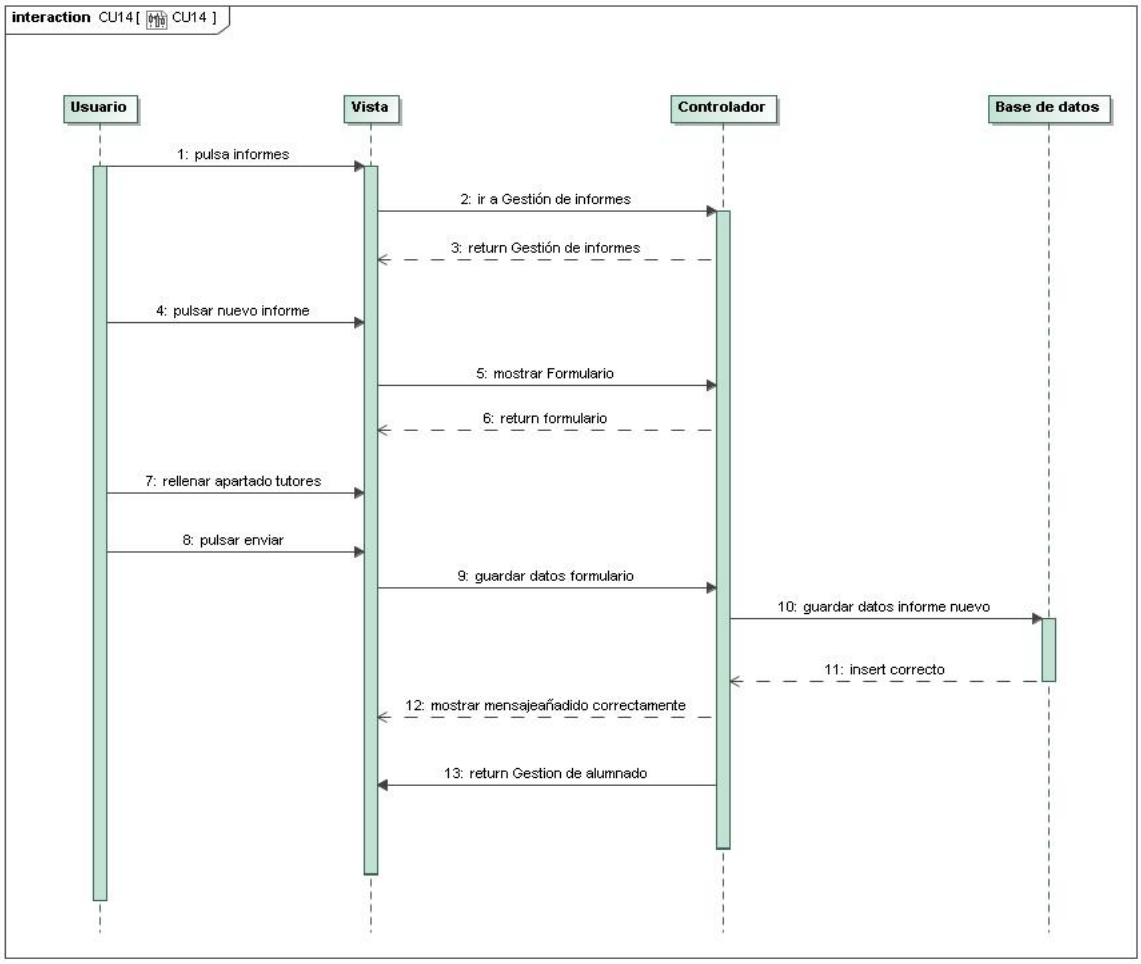

Figura 4.2.1.14 Caso de uso 14

4.2.1.15 Registrar la ausencia de un profesor

- **Precondición:** El usuario se encuentra en la página Gestión del Profesorado. Que el profesor se encuentre en la base de datos. Que el profesor anteriormente no estaba ausente.
- **Postcondición:** Que el profesor esté ausente.
- **Escenario principal:**
    1. El usuario busca el profesor cuyos datos quiere editar y hace click en el botón Editar.
    2. El sistema muestra el formulario con todos los campos del formulario rellenos.
    3. El usuario busca el campo Estado del profesor, lo cambia a ausente.
    4. El sistema muestra un cuadro de texto para rellenar en caso de que el profesor que va a estar ausente quiera dejar algún comentario para el profesor que lo sustituya.
    5. El usuario presiona editar.
    6. El sistema muestra un mensaje de que se ha modificado correctamente y vuelve a la página principal de Gestión del Profesorado.

- **Diagrama de secuencia:**

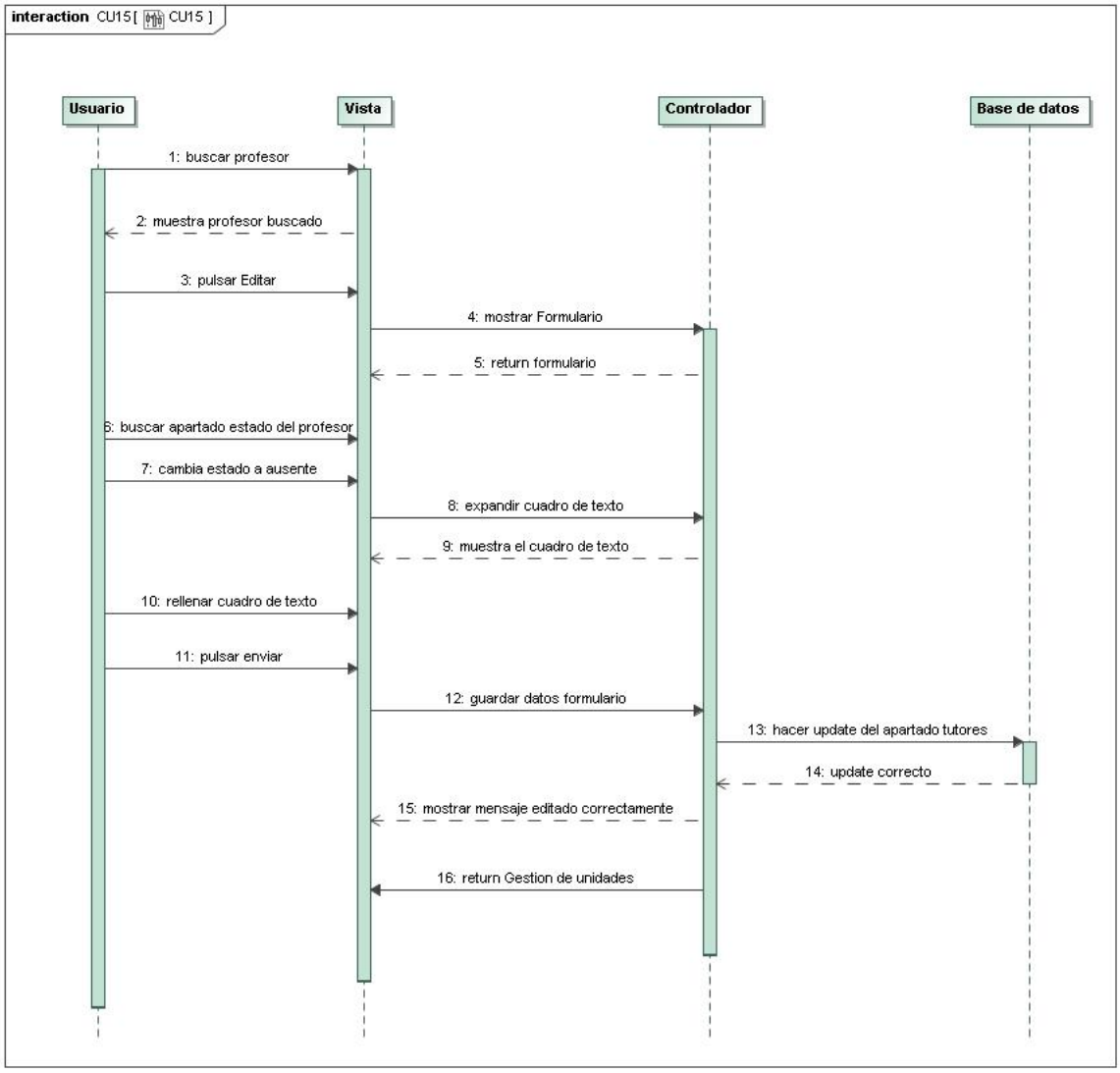

Figura 4.2.1.15 Caso de uso 15

4.2.1.16 Añadir un evento al calendario

- **Precondición:** El usuario se encuentra en la página Gestión de Eventos. El evento no se ha añadido previamente.
- **Postcondición:** El evento se ha añadido a la base de datos.
- **Escenario principal:**
    1. El usuario hace click en el botón Añadir.
    2. El sistema muestra el formulario con los campos que hay que rellenar.
    3. El usuario rellena los campos y presiona enviar.
    4. El sistema muestra un mensaje de que se ha añadido correctamente y vuelve a la página principal de Gestión de Eventos.

- **Diagrama de secuencia:**

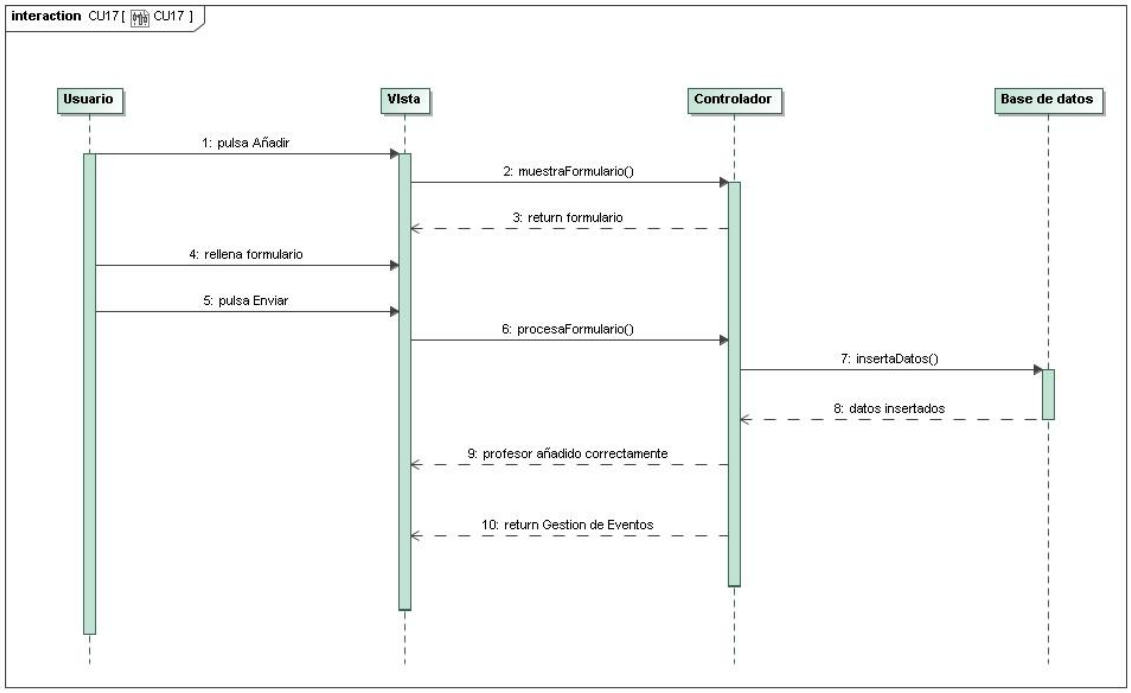

Figura 4.2.1.16 Caso de uso 16

4.2.1.17 Modificar los datos de un evento del calendario

- **Precondición:** El usuario se encuentra en la página Gestión de Eventos Los datos del evento que desean modificar se ha añadido previamente.
- **Postcondición:** Los datos del evento han sido modificados en la base de datos.
- **Escenario principal:**
    1. El usuario busca el título del evento cuyos datos quiere editar y hace click en el botón Modificar.
    2. El sistema muestra el formulario con todos los campos del formulario rellenos.
    3. El usuario modifica los campos que desee y presiona enviar.
    4. El sistema muestra un mensaje de que se ha modificado correctamente y vuelve a la página principal de Gestión de Eventos.

- **Diagrama de secuencia:**

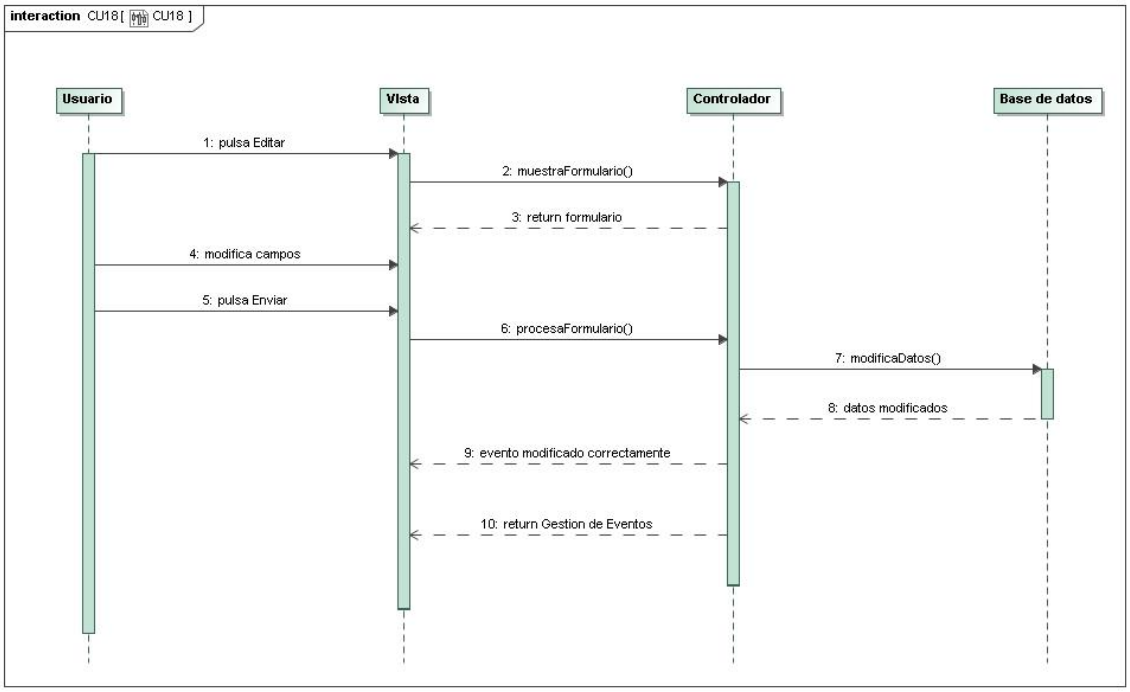

Figura 4.2.1.27 Caso de uso 17

4.2.1.18 Consultar un evento del calendario

- **Precondición:** El usuario se encuentra en la página Gestión de Eventos El evento cuyos datos se desean consultar ha sido añadido previamente.
- **Postcondición:** Los datos del evento se muestran correctamente.
- **Escenario principal:**
    1. El usuario busca el título del evento que quiere consultar y hace click en el botón consultar.
    2. El sistema muestra los datos del evento en cuestión.
- **Escenario alternativo:**
- **Diagrama de secuencia:**

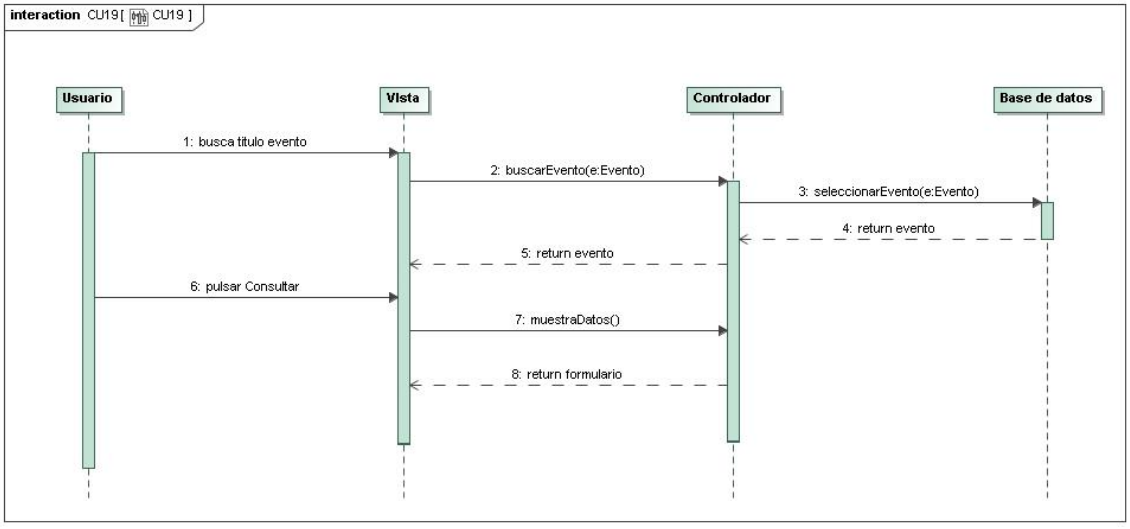

Figura 4.2.1.1 8 Caso de uso 18

4.2.1.19 Eliminar un evento del calendario

- **Precondición:** El usuario se encuentra en la página Gestión de Eventos El evento que se desea eliminar ha sido añadido previamente.
- **Postcondición:** El evento se ha eliminado de la base de datos.
- **Escenario principal:**
    1. El usuario busca el título del evento que quiere consultar y hace click en el botón eliminar.
    2. El sistema muestra un mensaje de que se ha eliminado correctamente y vuelve a la página principal de Gestión de Eventos.
- **Diagrama de secuencia:**

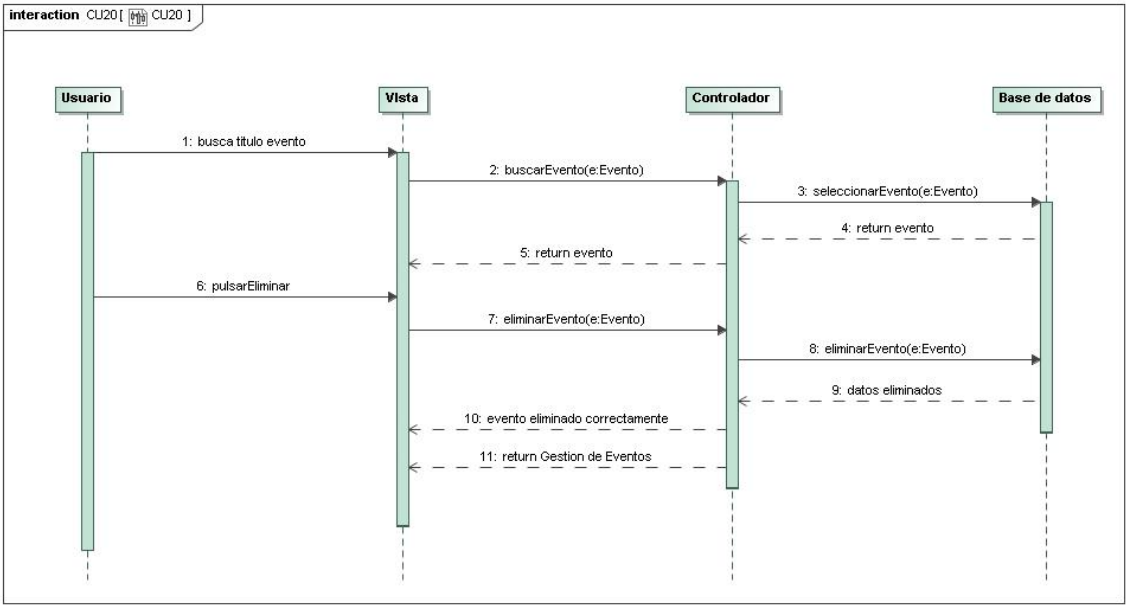

Figura 4.2.1.19 Caso de uso 19

4.2.1.20 Rellenar actas de ciclo

- **Precondición:** El usuario se encuentra en la página Informes y Actas.
- **Postcondición:** Que el sistema haya almacenado los datos rellenos en el informe.
- **Escenario principal:**
    1. El usuario pulsará en el botón Actas de ciclo.
    2. El sistema mostrará un listado con las actas de ciclo actuales y un botón Nueva acta para crear una nueva.
    3. El usuario pulsará el botón Nueva acta.
    4. El sistema le mostrará un formulario con los campos necesarios para rellenar.
    5. El usuario rellenará los campos y pulsará el botón enviar.
    6. El sistema mostrará un mensaje de que se ha guardado correctamente el informe y volverá a la página principal de Informes y Actas.
- **Diagrama de secuencia:**

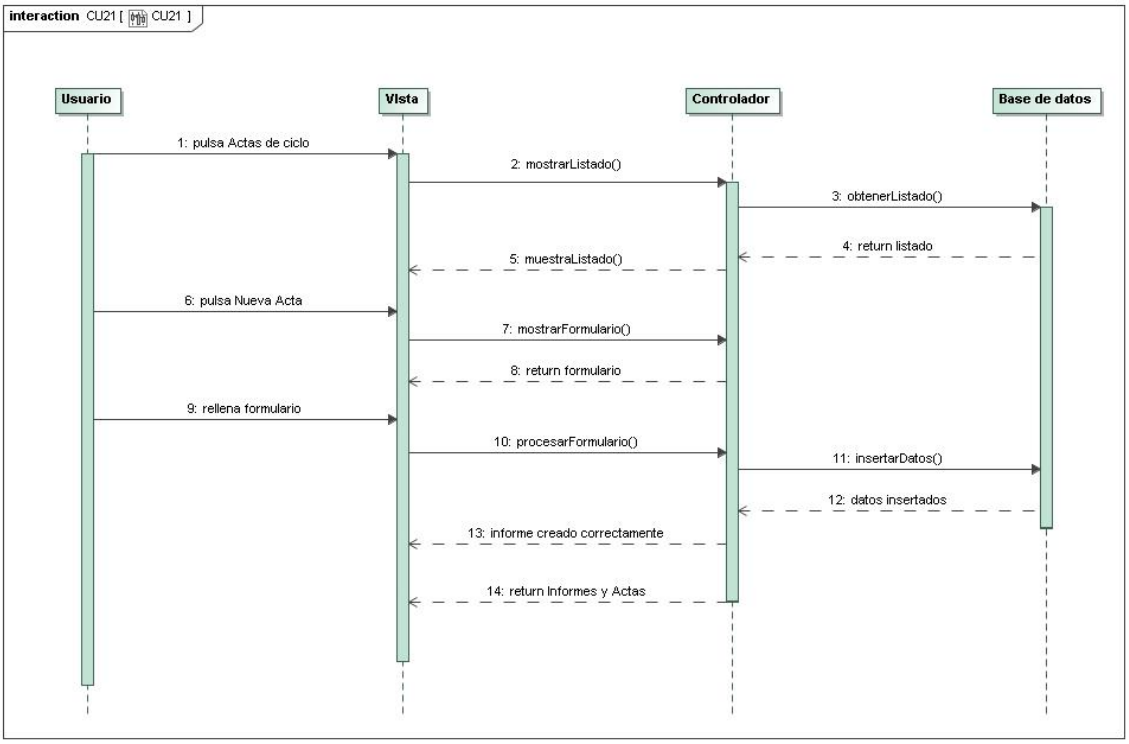

Figura 4.2.1.20 Caso de uso 20

4.2.1.21 Rellenar actas de incidencias TIC

- **Precondición:** El usuario se encuentra en la página Informes y Actas.
- **Postcondición:** Que el sistema haya almacenado los datos rellenos en el informe.
- **Escenario principal:**
    1. El usuario pulsará en el botón Actas de incidencias.
    2. El sistema mostrará un listado con las actas de ciclo actuales y un botón Nueva acta para crear una nueva.
    3. El usuario pulsará el botón Nueva acta.
    4. El sistema le mostrará un formulario con los campos necesarios para rellenar.
    5. El usuario rellenará los campos y pulsará el botón enviar.
    6. El sistema mostrará un mensaje de que se ha guardado correctamente el informe y volverá a la página principal de Informes y Actas.
- **Diagrama de secuencia:**

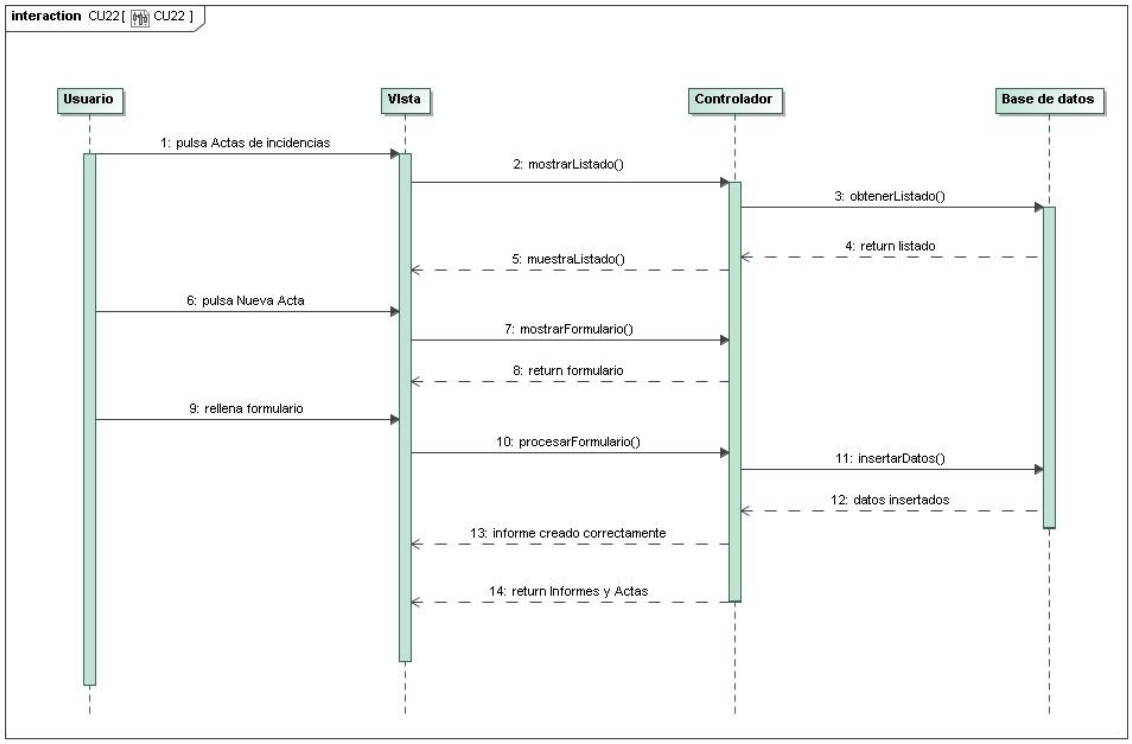

Figura 4.2.1.21 Caso de uso 21

5

Implementación

En este capítulo se detallará la implementación de la aplicación web, la cual ha constado de 3 iteraciones completas y funcionalmente independientes en las que, además, se ha ido rediseñando la base de datos en función del módulo que se desarrollaba en cada momento.

A continuación se mostrará una imagen de las tecnologías utilizadas en cada parte del desarrollo de esta aplicación web que sigue el patrón MVC. Se ha decidido utilizar estas tecnologías ya que eran conocidas por el programador, permiten realizar una web modular y estructurada, y las ha utilizado anteriormente, además tienen librerías que han resultado muy útiles en ciertas partes de la herramienta como puede ser la creación de un PDF con el informe del alumno o el envío de correos a los profesores que deben rellenar ese informe a modo de aviso.

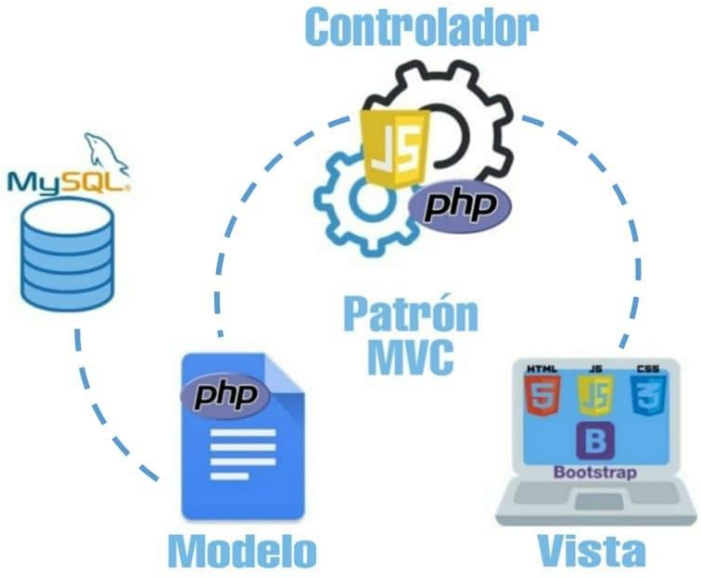

Figura 5.1 Diagrama con las tecnologías utilizadas

5.1 Primera iteración

En esta iteración se ha realizado el diseño inicial de la base de datos, ya descrita completamente en el apartado 4.1, con las entidades necesarias y los datos básicos que se iba a introducir para cada sección de la aplicación. Además, se realizaron los casos de uso que se utilizarán para implementar los requisitos (apartado 4.2.1), se desarrollaron los formularios de inicio de sesión, de registro, la página principal y los importadores de archivos CSV, uno para cada sección que sea necesario, los cuales se añaden en la próxima iteración en su apartado de gestión correspondiente.

A continuación se describirá en distintos apartados cada uno de los desarrollos realizados en esta iteración, se incluirán imágenes de código y del diseño final del formulario.

5.1.0 Conexión con la base de datos

Se encuentra en el archivo *BDConnection.php* ubicado en la carpeta *database*. El propósito de este archivo es crear la conexión con la base de datos y los parámetros necesarios de forma que, cada vez que sea necesario utilizarla para realizar algún tipo de consulta no haya que crear los parámetros y la conexión de forma reiterada, solo con incluir el archivo ya se cree la conexión y se puedan utilizar las variables de ese archivo para realizar las consultas necesarias. Este archivo se importará mediante la función *require* siempre que sea necesario realizar algún tipo de uso de la base de datos.

El código PHP que crea la conexión con la base de datos es el siguiente:

```php
$hostname = "localhost"; //servidor
$dbname = "centroeducativo"; //nombre de la base de datos
$username = "root"; //usuario
$password = "ROOT"; //contraseña
$connect = mysqli_connect($hostname, $username, $password, $dbname);
```

Figura 5.1.0.1 Código PHP de la conexión con la base de datos

Además, en este archivo también se encuentra la URL raíz del proyecto, que se utilizará más adelante para redireccionar, y las variables *correo* y *correopwd*, que contienen el usuario y contraseña de una cuenta de correo creada que se utilizará en el apartado 5.3.2 para enviar correos a los profesores que den clase a un alumno cuyo informe de tutoría se ha creado.

```php
$url = "http://localhost/CentroEducativo/"; //contiene la direccion base del proyecto

$correo = "centroeducativotfg@gmail.com"; //email creado para poder enviar correos
$correopwd = "cetfg2021"; //contraseña del email
```

Figura 5.1.0.2 Código PHP de la URL y cuenta de correo

5.1.1 Inicio de sesión

Esta página web es la primera que aparece en cuanto se ejecuta el proyecto, se llama *index.php* y se trata de un formulario básico de inicio de sesión. En él se pide introducir el correo del usuario, un ejemplo sería *profesor@colegio.com*, y la contraseña justo debajo y pulsar el botón *Acceder*.

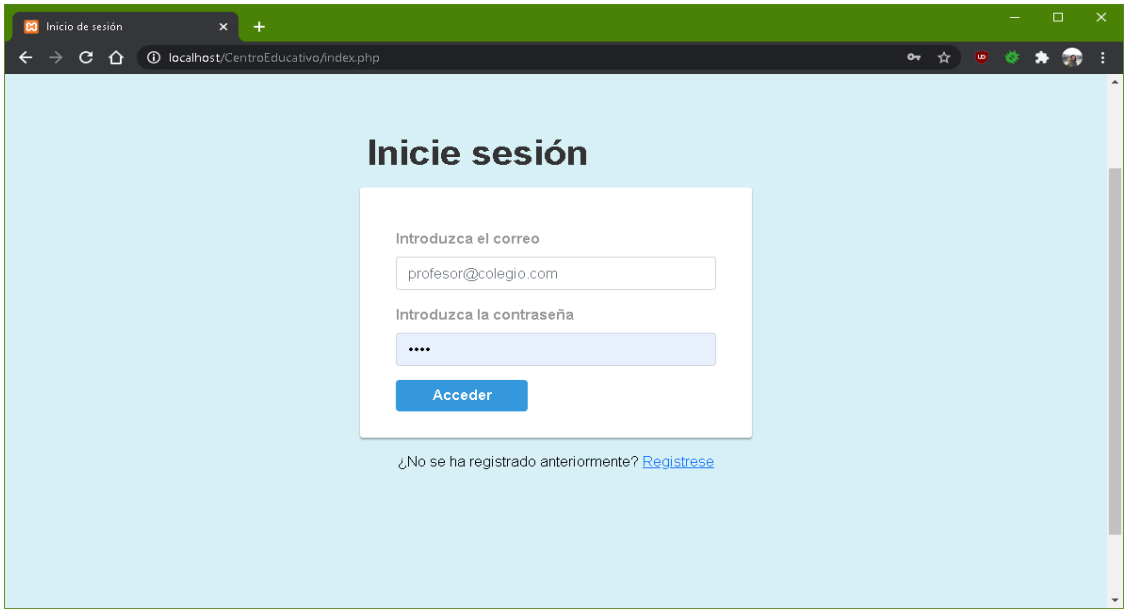

Figura 5.1.1.1 Formulario de inicio de sesión

En caso de que el usuario no se haya registrado aún, bajo el formulario aparece un enlace que lleva a la página *register.php* la cual se detallará en el apartado 5.1.2. El código HTML del formulario es el siguiente:

```
<!-- formulario de inicio de sesion -->
<div class="signup-form" style="margin-top:200px">
    <h2 class="h1" > Inicie sesi&oacute;n </h2><br>
    <form class="row g-3" method="POST" action="lib/login.php" enctype="multipart/form-data">
        <div class="col-12">
            <label for="email" class="form-label"><b>Introduzca el correo</b></label>
            <input type="text" placeholder="profesor@colegio.com" name="email" required class="form-control"/>
        </div>
        <div class="col-md-12">
            <label for="psw" class="form-label"><b>Introduzca la contraseña</b></label>
            <input type="password" placeholder="contraseña" name="pass" required class="form-control"/>
        </div>
        <!-- boton submit -->
        <div class="col-12">
            <button type="submit" class="btn btn-primary">Acceder</button>
        </div>
    </form>
    <!-- si no se ha registrado antes, el formulario te lleva al formulario de registro -->
    <div class="hint-text">&iquest;No se ha registrado anteriormente&#63;
        <a href="<?php echo $url;?>register.php" name="register">Registrese</a>
    </div>
</div>
```

Figura 5.1.1.2 Código HTML del formulario de inicio de sesión

En cuanto se pulsa el botón, los datos son enviados mediante el método POST a una página llamada *login.php* que se encuentra en la carpeta *lib* del proyecto. Lo primero que hace es iniciar la sesión y cargar el archivo *BDConnection.php* descrito en el apartado 5.1.0.

```php
<?php
session_start(); //inicia la sesión
require('../database/BDConnection.php'); //carga el enlace con la base de datos
?>
```

Figura 5.1.1.3 Código PHP inicio de sesión y carga de fichero

Tras esto, se accede a la base de datos y se realiza una consulta SELECT en la que se obtiene todos los usuarios que se encuentran registrados en la base de datos y se realiza una búsqueda comprobando si alguno de los usuarios registrados coincide en correo con el que ha puesto el usuario que está tratando de iniciar sesión y que la contraseña introducida coincide con la contraseña que hay en la base de datos, para ello se usa la función de PHP *password_verify()* ya que la contraseña que se encuentra en la base de datos está cifrada. Además obtiene la id que pertenece al usuario que ha iniciado sesión para pasarla más adelante por URL a la página principal.

```php
$lUsers = mysqli_query($connect, "SELECT * FROM login") or die(mysqli_connect_error());
$row = mysqli_fetch_assoc($lUsers);
$found = 0;

while ($found == 0 && $row != NULL) {
    //comprueba que el usuario ya se ha registrado
    if (strcasecmp($row['email'], $_POST['email'] == 0) && password_verify($_POST['pass'], $row['password'])) {
        $found = 1;
        $id = $row['id_login'];
    } else {
        $row = mysqli_fetch_assoc($lUsers);
    }
}
```

Figura 5.1.1.4 Código PHP que busca entre los usuarios registrados

En caso de que coincidan, la página redirige a la página principal *home.php* (apartado 5.1.3), si no coincide el sistema mostrará un mensaje advirtiendo al usuario de que el correo o la contraseña no coincide con ningún usuario que se haya registrado anteriormente en la base de datos y le insta a introducir un correo y contraseña correctos.

```php
if ($found == 1) { //como se ha registrado, va a la página de inicio
    $urlr = $url."home.php?id_login=".$id;
    header('Location: ' . $urlr);
    exit();
} else { //como no se ha registrado, vuelve al formulario de iniciar sesión
    echo "<script>
    alert('El correo o la contraseña no coinciden con nigún usuario en la base de datos, "
    ."por favor introzca un correo o contraseña correctos.');history.back();</script>";
}
```

Figura 5.1.1.5 Código PHP que permite iniciar sesión

5.1.2 Registro

Si el usuario accede por primera vez a la web, será necesario que se registre. Para ello debe pulsar en el enlace llamado *Regístrese*, que se encuentra debajo del formulario de inicio de sesión, que lleva al usuario a la página *register.php*. En esta página el usuario debe poner su nombre de usuario, correo y contraseña y pulsar el botón *Registrarse*.

Para registrarse, el usuario deberá utilizar el mismo correo que proporcionó en sus datos personales, si desea cambiarlo deberá ir a sus datos correspondientes y cambiar la información que ha proporcionado. Cuando lo haga, también se actualizará el correo con el que inicia sesión al nuevo correo que ha proporcionado.

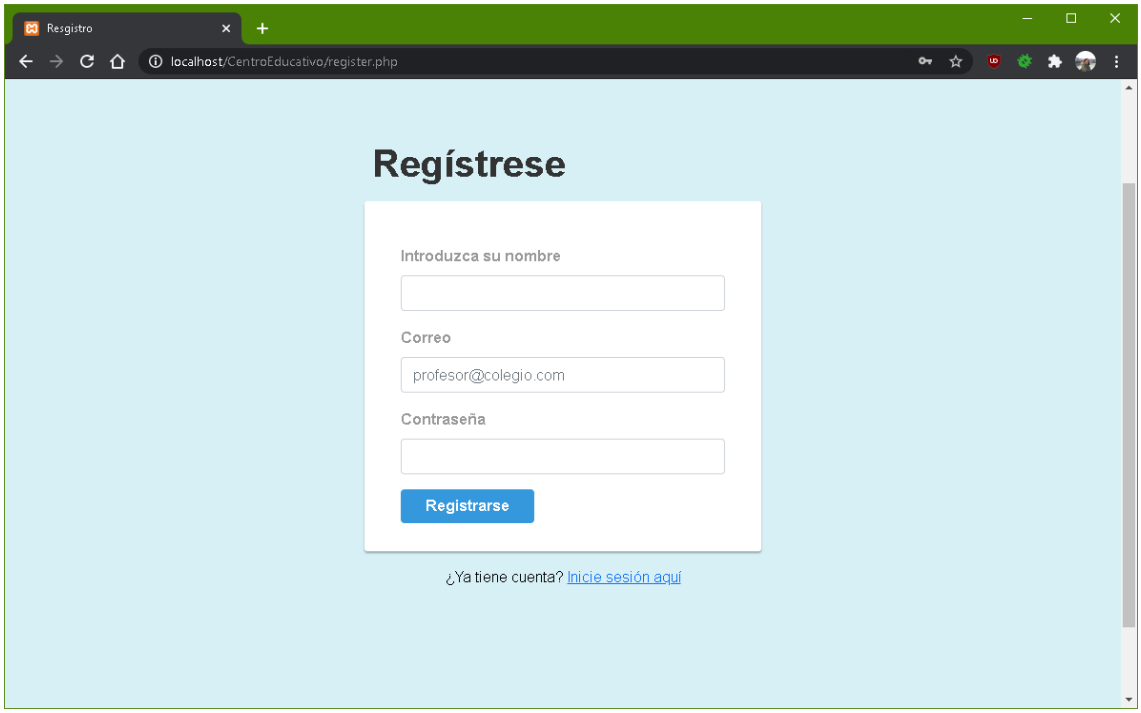

Figura 5.1.2.1 Formulario de registro

En caso de que el usuario ya se haya registrado, en la parte posterior del formulario de registro, se encuentra un enlace llamado *Inicie sesión aquí* que lleva al usuario al formulario de inicio de sesión (apartado 5.1.1). El código HTML del formulario de registro es el siguiente:

```
<!-- formulario de registro -->
 <div class="signup-form" style="margin-top:200px">
    <h2 class="h1"> Regístrese </h2><br>
    <form class="row g-3" method="POST" action="lib/saveRegister.php" enctype="multipart/form-data">
        <div class="col-12">
            <label for="nombre" class="form-label"><b>Introduzca su nombre</b></label>
            <input type="text" name="nombre" class="form-control" required/>
        </div>
        <div class="col-md-12">
            <label for="uname" class="form-label"><b>Correo</b></label>
            <input type="email" placeholder="profesor@colegio.com" name="email" class="form-control" required/>
        </div>
        <div class="col-md-12">
            <label for="psw" class="form-label"><b>Contraseña</b></label>
            <input type="password" name="pass" class="form-control" required/>
        </div>
        <!-- boton submit -->
        <div class="col-12">
            <button type="submit" class="btn btn-primary">Registrarse</button>
        </div>
    </form>
    <!-- si se ha registrado antes, el enlace lleva al formulario de inicio de sesion -->
    <div class="hint-text">¿Ya tiene cuenta? <a href="<?php echo $url;?>index.php">Inicie sesi&oacute;n aqu&iacute;</a></div>
 </div>
```

Figura 5.1.2.2 Código HTML del formulario de registro

En cuanto el usuario pulsa el botón, los datos son enviados mediante el método POST a una página llamada *saveRegister.php* que se encuentra en la carpeta *lib* del proyecto. Lo primero que hace es cargar el archivo *BDConnection.php* descrito en el apartado 5.1.0, realizar dos consulta SELECT, una para seleccionar todos los usuarios registrados en la tabla *login* y otra para obtener todos los profesores que se han insertado en la aplicación. Tras esto, realiza una búsqueda para comprobar si alguno de estos usuarios se ha dado de alta con un correo que coincida con el correo que el nuevo usuario ha escrito en el formulario.

```php
<?php require('../database/BDConnection.php'); ?>

<?php
//selecciona todos los datos de la tabla login
$lUsers = mysqli_query($connect, "SELECT * FROM login") or die(mysqli_connect_error());
$row = mysqli_fetch_assoc($lUsers);
//selecciona todos los datos de la tabla profesor
$lprof = mysqli_query($connect, "SELECT * FROM profesor") or die(mysqli_connect_error());
$p = mysqli_fetch_assoc($lprof);
$found = 0; //0 si no hay nadie registrado en la base de datos con ese correo, 1 si lo hay
$foundp = 0; //0 si no hay ningún profesor en la base de datos con ese correo, 1 si lo hay
//comprueba si ya hay alguien registrado con ese correo
while ($found == 0 && $row != NULL) {
    if (strcasecmp($row['email'], $_POST['email']) == 0) { //si en la base de datos ya esta el correo
        $found = 1; //pone found a 1
    } else { //si no está el correo sigue buscando hasta que lo encuentra o no hay más donde buscar
        $row = mysqli_fetch_assoc($lUsers);
    }
}
```

Figura 5.1.2.3 Código PHP de búsqueda de coincidencias en login

A continuación comprueba si alguno de los profesores que se han añadido a la base de datos, en caso de que esté, se guarda el DNI en una variable para posteriormente actualizar el *id_login* del profesor y así enlazarlo con su inicio de sesión.

```php
//comprueba si hay un profesor en la bd con ese correo para relacionarlo con este login
while ($foundp == 0 && $p != NULL) {
    if (strcasecmp($p['email'], $_POST['email']) == 0) { //si en la base de datos ya esta el correo
        $foundp = 1; //pone found a 1
        $dni = $p['dni_profesor'];
    } else { //si no está el correo sigue buscando hasta que lo encuentra o no hay más donde buscar
        $p = mysqli_fetch_assoc($lprof);
    }
}
```

Figura 5.1.2.4 Código PHP búsqueda en login

Tras esto, busca en la tabla unidad si hay algún tutor que coincida en nombre y apellidos con el usuario que ha iniciado sesión. En caso de que lo haga, pondrá la variable *foundu* a 1 y, más adelante, si esta variable está a 1 se guarda en la variable *rol* los números 2 y 3 que pertenecen a los roles de profesor y tutor respectivamente. Si la variable está a 0, solo se guarda el número 2 que es el rol de profesor.

```php
//comprueba si el profesor es tutor de algún curso, si lo es se pone de rol profesor y tutor
//si no lo es solo se pone de rol profesor
$qu = mysqli_query($connect, "SELECT * FROM unidad");
$u = mysqli_fetch_assoc($qu);
$foundu = 0; //se pone a 1 si el tutor coincide con la persona que inició sesion
while ($foundu == 0 && $u != NULL) {
    if (strpos($u['tutor'], $p['profesor']) !== false) {
        $foundu = 1;
    } else {
        $u = mysqli_fetch_assoc($qu);
    }
}
//si es tutor, se pone los roles 2 y 3 (profesor y tutor)
if($foundu == 1){
    $rol ="2 3";
}else{ //si no es tutor, solo se pone el rol 2 (profesor
    $rol="2";
}
```

Figura 5.1.2.5 Código PHP búsqueda en unidad

Si no hay ningún usuario registrado con el correo enviado por método POST, se cifra la contraseña utilizando la función *password_hash()* de PHP, insertan los datos en la base de datos y se obtiene la id de la inserción para poder enviarla a la página de inicio mediante la URL. Una vez realizada la consulta, si hay un profesor en la base de datos con el mismo correo que se ha registrado, se asocia el id de haber iniciado sesión a sus datos personales.

Para terminar, la página redireccionará a la página principal *home.php* que se detallará en el apartado 5.1.3. En caso de que ya haya otro usuario con ese correo, el sistema mostrará un mensaje al usuario informándole del error y le instará a introducir otro correo.

```php
if ($found == 0) { //si no hay nadie registrado con ese correo, se registra y entra en la web
    //el rol de usuario es el numero 2 que pertenece al profesor en la tabla roles
    $query = sprintf('INSERT INTO login (email, password, nombre, rol) VALUES ("%s", "%s", "%s", "%s")',
            $_POST['email'], password_hash($_POST['pass'], PASSWORD_DEFAULT), $_POST['nombre'], $rol); //consulta
    echo $res = mysqli_query($connect, $query) or die(mysqli_connect_error() . 'error al insertar');
    $id = mysqli_insert_id($connect); //obtiene la id del utlimo elemento que se ha insertado
    $urlr = $url . "home.php?id_login=" . $id;

    if ($foundp == 1) { //si hay un profesor en la base de datos con ese correo
        $q = "UPDATE profesor SET id_login='" . $id . "' WHERE dni_profesor='" . $dni . "'";
        echo $q;
        echo $rp = mysqli_query($connect, $q)or die(mysqli_connect_error() . 'error al update profesor id_login');
    }

    header('Location: ' . $urlr); //redirecciona a la url
} else { //si ya hay alguien con ese correo, vuelve a la pagina de registrarse para que ponga otro correo
    echo '<script> alert("El correo coincide con el de otra persona, por favor utilice otro correo para registrarse."); '
    . 'history.back();</script>';
}
```

Figura 5.1.2.6 Código PHP de registro y redirección

5.1.3 Página principal

Es la página a la que accede el usuario una vez que se ha registrado o ha iniciado sesión.

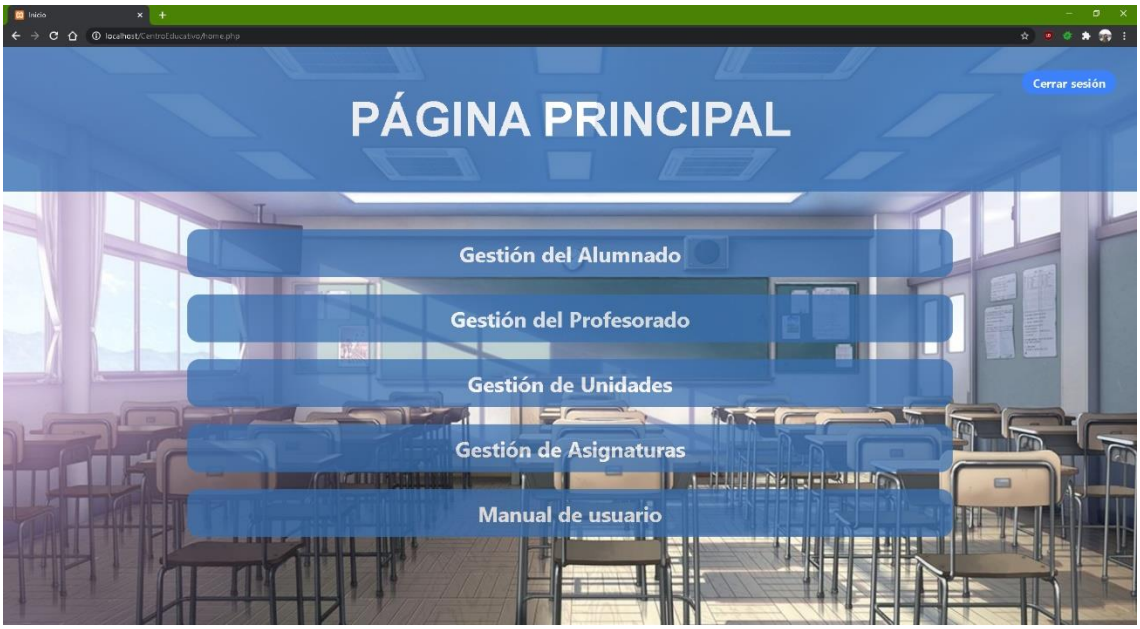

Figura 5.1.3.1 Página principal home.php

Esta página posee un botón para cada sección de gestión, las cuales se explicarán en la segunda (apartado 5.2) y tercera (apartado 5.3) iteración. Además de un botón que muestra el manual de usuario en caso de que sea necesaria su consulta por algún usuario, también posee un botón llamado *Cerrar sesión* que redirige al usuario a la página *login.php*.

```
<div class="text-center neg">
    <h1>Página principal</h1>
</div><br><br>
<div class="container-c text-center">
    <button role="link" class="btn btn-secondary" onclick="window.location = '<?php echo $url;?>index.php'">Cerrar sesi&oacute;n</button>
</div>
<div class="col container text-center align-items-center" style="line-height:110px">
    <div class="">
        <button role="link" class="btn btn-primary" onclick="window.location = '<?php echo $url;?>galumno.php?id_login=<?php echo $_GET['id_login']; ?>'">Gesti&oacute;n del Alumnado</button>
    </div>
    <div class="">
        <button role="link" class="btn btn-primary" onclick="window.location = '<?php echo $url;?>gprofesor.php?id_login=<?php echo $_GET['id_login']; ?>'">Gesti&oacute;n del Profesorado</button>
    </div>
    <div class="">
        <button role="link" class="btn btn-primary" onclick="window.location = '<?php echo $url;?>gunidad.php?id_login=<?php echo $_GET['id_login']; ?>'">Gesti&oacute;n de Unidades</button>
    </div>
    <div class="">
        <button role="link" class="btn btn-primary" onclick="window.location = '<?php echo $url;?>gasignatura.php?id_login=<?php echo $_GET['id_login']; ?>'">Gesti&oacute;n de Asignaturas</button>
    </div>
    <div class="">
        <button role="link" class="btn btn-primary" onclick="window.location = '<?php echo $url;?>gmanual.php?id_login=<?php echo $_GET['id_login']; ?>'">Manual de usuario</button>
    </div>
</div>
```

Figura 5.1.3.2 Código HTML de la página principal

5.1.4 Importador de alumnos de una unidad

La finalidad de esta sección de código contenida en *uploadAlumno.php*, del que se hará uso en la sección de Gestión del alumnado (apartado 5.3.1), es comprobar que el formato del fichero es el adecuado, recoger todos los datos que vienen en él y añadirlos a la entidad *Alumno* de la base de datos. A continuación se detallará paso por paso cada parte de este código.

Lo primero que hace es cargar el archivo *BDConnection.php* y asegurarse de que no hay errores al cargarlo. Si no hay errores, se asegura de que el formato del fichero es CSV. En el caso de que no lo sea, vuelve a la página principal de Gestión del alumnado y se muestra un mensaje de error de que el fichero no tiene el formato adecuado.

```php
<?php require('../database/BDConnection.php'); ?>

<?php
//comprueba que el fichero se cargó correctamente
if (isset($_FILES['file']) && $_FILES['file']['error'] === UPLOAD_ERR_OK) {
    $fileName = $_FILES['file']['name'];
    $fileComps = explode(".", $fileName); //partes del nombre del fichero
    $ext = strtolower(end($fileComps)); //extension del archivo
    if (strcmp($ext, "csv") != 0) { //comprueba que el formato del archivo es el adecuado
        //mensaje de error y vuelta ala página de carga
        $urlr = $url."galumno.php?ft=0&id_login=" . $_GET['id_login'];
        header('Location: ' . $urlr);
        exit();
    }
}
```

Figura 5.1.4.1 Comprobación de errores del fichero

Si no hubo ningún error ni en la carga ni en el formato del archivo, se procede a abrirlo con permisos de lectura y se obtiene la primera línea de este. La línea se formatea para que tenga codificación UTF-8, pero no es necesario agregarla a la base de datos ya que solo posee información sobre los campos que vienen en el fichero.

Una vez leída la primera, puede proceder a leer la segunda en la cual ya viene información que hay que introducir. En el caso de que haya datos, es decir, que el archivo no esté vacío, se obtiene la *unidad* a la que pertenece el alumno que se va a introducir y obtiene un listado de alumnos que pertenece a la misma unidad que el alumno que se va introducir.

```php
$archive = fopen($_FILES['file']['tmp_name'], "r"); //abre el fichero
$line = utf8_encode(fgets($archive)); //obtiene la primera linea del fichero
$data = fgetcsv($archive); //obtiene la segunda linea

if ($data !== false) {
    $unidad = utf8_encode($data[14]);  //obtiene la unidad del alumno
    //obtiene todos los alumnos de la base de datos que pertenecen a la unidad
    $un = mysqli_query($connect, "SELECT * FROM alumno WHERE unidad='" . $unidad . "'") or die(mysqli_connect_error());
```

Figura 5.1.4.2 Apertura del fichero y obtención de primeros datos

Si no hay alumnos introducidos anteriormente en la unidad a la que pertenece el nuevo alumno, entra en el bucle y obtiene el *id escolar* del alumno y crea una variable *found* que la inicializa a 0. Además obtiene todos los alumnos ya introducidos en la base de datos.

El sistema realiza una búsqueda y comprueba si el *id escolar* del alumno que se va a introducir coincide con otro *id escolar* anteriormente introducido. Esto no debería ser posible ya que el *id escolar* es único para cada alumno, por lo que si coincide es que el alumno ya ha sido introducir previamente. Si se da este caso, se pone la variable *found* a 1.

```php
if (mysqli_num_rows($un) == 0) { //si no hay alumnos en la unidad
    while ($data !== false) {
        $id = utf8_encode($data[2]); //obtiene el id escolar del alumno
        $foundA = 0; //esta a 0 si el alumno no está en la base de datos, a 1 en el caso de que ya esté
        //obtiene todos los alumnos
        $all = mysqli_query($connect, "SELECT * FROM alumno") or die(mysqli_connect_error());
        $rowa = mysqli_fetch_assoc($all);

        //comprueba si el alumno no se ha introducido anteriormente
        while ($foundA == 0 && $rowa != NULL) { //comprueba si ya hay alguien con ese id escolar
            if (strcasecmp($rowa['id_escolar'], $id) == 0) { //si exite en la tabla
                $foundA = 1; //marca found a 1
            } else { //si no existe
                $rowa = mysqli_fetch_assoc($all); //coge el siguiente alumno de la base de datos
            }
        }
```

Figura 5.1.4.3 Búsqueda en la base de datos

Si el alumno no se ha introducido anteriormente, es decir el valor de *found* sigue siendo 0, se guarda en cada variable un dato del alumno que se va a introducir para hacer posteriormente la consulta INSERT.

```php
//si no está en la base de datos
if ($foundA == 0) {
    //guarda en una variable cada parte del csv
    //cada dato se codifica a utf8
    $alum = utf8_encode($data[0]);
    $estado = utf8_encode($data[1]);
    $dni = utf8_encode($data[3]);
    $direc = utf8_encode($data[4]);
    $cdpostal = utf8_encode($data[5]);
    $loc_resi = utf8_encode($data[6]);
    //formatea la fecha para que no haya
    //problemas al introducirla en la BD
    $fecha_nac = date("Y-m-d",
    strtotime(str_replace('/', '-', $data[7])));
    $prov_resi = utf8_encode($data[8]);
    $tlf = utf8_encode($data[9]);
    $tlf_urg = utf8_encode($data[10]);
    $email = utf8_encode($data[11]);
    $curso = utf8_encode($data[12]);
    $num_exp = utf8_encode($data[13]);
    $apellido1 = utf8_encode($data[15]);
    $apellido2 = utf8_encode($data[16]);
    $nombre = utf8_encode($data[17]);
    $dni_tutor1 = utf8_encode($data[18]);
    $apellido1_tutor1 = utf8_encode($data[19]);
    $apellido2_tutor1 = utf8_encode($data[20]);
    $nombre_tutor1 = utf8_encode($data[21]);
    $sexo_tutor1 = utf8_encode($data[22]);
    $dni_tutor2 = utf8_encode($data[23]);
    $apellido1_tutor2 = utf8_encode($data[24]);
    $apellido2_tutor2 = utf8_encode($data[25]);
    $nombre_tutor2 = utf8_encode($data[26]);
    $sexo_tutor2 = utf8_encode($data[27]);
    $loc_nac = utf8_encode($data[28]);
    $anyo_mat = utf8_encode($data[29]);
    $num_mats = utf8_encode($data[30]);
    $observaciones = utf8_encode($data[31]);
    $prov_nac = utf8_encode($data[32]);
    $pais_nac = utf8_encode($data[33]);
    $edad_fin_anyo = utf8_encode($data[34]);
    $nacionalidad = utf8_encode($data[35]);
    $sexo = utf8_encode($data[36]);
    //formatea la fecha para que no haya
    //problemas al introducirla en la BD
    $fecha_mat = date("Y-m-d",
    strtotime(str_replace('/', '-', $data[37])));
    $num_ss = utf8_encode($data[38]);
```

Figura 5.1.4.4 Almacenamiento de datos en variables

Una vez guardados los datos, se realiza la consulta INSERT para almacenar los datos del alumno en la base de datos.

```
$query = sprintf('INSERT INTO alumno (id_escolar, alumno, estado_matricula, dni_alumno, direccion, '
        . 'codigo_postal, localidad, fecha_nacimiento, provincia, telefono, tlf_emergencia, email,'
        . 'curso, num_expediente, unidad, apellido1, apellido2, nombre, dni_tutor1, apellido1_tutor1, '
        . 'apellido2_tutor1, nombre_tutor1, sexo_tutor1, dni_tutor2, apellido1_tutor2,'
        . 'apellido2_tutor2, nombre_tutor2, sexo_tutor2, localidad_nacimiento, anyo_matricula, '
        . 'num_matriculas_curso_actual, observaciones_matricula, provincia_nacimiento, pais_nacimiento, '
        . 'edad_final_anyo_matricula, nacionalidad, sexo, fecha_matricula, num_seg_social) '
        . 'VALUES ("%s", "%s", "%s", "%s", "%s", "%s", "%s", "%s","%s", "%s", "%s", "%s", "%s", "%s", "%s",'
        . '"%s", "%s", "%s", "%s", "%s", "%s", "%s", "%s", "%s", "%s", "%s", "%s", "%s", "%s", "%s", '
        . '"%s", "%s", "%s", "%s", "%s", "%s", "%s", "%s")',
        $id, $alum, $estado, $dni, $direc, $cdpostal, $loc_resi, $fecha_nac, $prov_resi, $tlf, $tlf_urg,
        $email, $curso, $num_exp, $unidad, $apellido1, $apellido2, $nombre, $dni_tutor1,
        $apellido1_tutor1, $apellido2_tutor1, $nombre_tutor1, $sexo_tutor1, $dni_tutor2, $apellido1_tutor2,
        $apellido2_tutor2, $nombre_tutor2, $sexo_tutor2, $loc_nac, $anyo_mat, $num_mats, $observaciones,
        $prov_nac, $pais_nac, $edad_fin_anyo, $nacionalidad, $sexo, $fecha_mat, $num_ss);

$res = mysqli_query($connect, $query) or die(mysqli_connect_error() . " error al importar fichero alumnos");
```

Figura 5.1.4.5 Inserción de datos

En el caso de que el alumno ya estuviera en la base de datos, el sistema cierra el fichero y vuelve a la página principal de Gestión del alumnado y se muestra un mensaje de error de que no se pudo cargar el fichero porque el alumno ya estaba en la base de datos.

```
} else { //si el alumno está en la bd
    fclose($archive); //cierra el fichero
    $urlr = $url."galumno.php?ok=0&id_login=" . $_GET['id_login'];
    header('Location: ' . $urlr); //redirecciona
    exit();
}
```

Figura 5.1.4.6 Caso de que el alumno se haya introducido anteriormente

En el caso de que hubiera alumnos introducidos cuya unidad es igual a la del alumno que se va a introducir, el sistema cierra el fichero y vuelve a la página principal de Gestión del alumnado y se muestra un mensaje de error de que no se pudo cargar el fichero porque hay alumnos en dicha unidad.

```
} else { //ai hay alumnos en la unidad
    fclose($archive); //cierra el fichero
    $urlr = $url."galumno.php?oki=0&id_login=" . $_GET['id_login'];
    header('Location: ' . $urlr); //redirecciona
    exit();
}
```

Figura 5.1.4.7 Caso de que haya alumnos en la misma unidad

Si todo va bien y se ha podido insertar el alumno correctamente, el sistema cierra el fichero y vuelve a la página principal de Gestión del alumnado y se muestra un mensaje de que se ha cargado el fichero correctamente.

```
    fclose($archive); //cierra el fichero
    $urlr = $url."galumno.php?ok=1&id_login=" . $_GET['id_login'];
    header('Location: ' . $urlr); //redirecciona
```

Figura 5.1.4.8 Redirección a Gestión del alumnado

5.1.5 Importador de asignaturas

La finalidad de esta sección de código contenida en *uploadMatUnidProf.php*, del que se hará uso en la sección de Gestión de asignaturas (apartado 5.2.3), es comprobar que el formato del fichero es el adecuado, recoger todos los datos que vienen en él y añadirlos a la entidad *materia* de la base de datos. A continuación se detallará paso por paso cada parte de este código.

Lo primero que hace es cargar el archivo *BDConnection.php* y asegurarse de que no hay errores al cargarlo. Si no hay errores, se asegura de que el formato del fichero es CSV. En el caso de que no lo sea, vuelve a la página principal de Gestión de asignaturas y se muestra un mensaje de error de que el fichero no tiene el formato adecuado.

```php
<?php require('../database/BDConnection.php'); ?>

<?php
//comprueba que el fichero se cargó correctamente
if (isset($_FILES['file']) && $_FILES['file']['error'] === UPLOAD_ERR_OK) {
    $fileName = $_FILES['file']['name'];
    $fileComps = explode(".", $fileName); //partes del nombre del fichero
    $ext = strtolower(end($fileComps)); //extension del archivo
    if (strcmp($ext, "csv") != 0) { //comprueba que el formato del archivo es el adecuado
        //mensaje de error y vuelta ala página de carga
        $urlr = $url."gasignatura.php?ft=0&id_login=" . $_GET['id_login'];
        header('Location: ' . $urlr);
        exit();
    }
}
```

Figura 5.1.5.1 Comprobación de errores del fichero

Si no hubo ningún error ni en la carga ni en el formato del archivo, se procede a abrirlo con permisos de lectura y se obtiene la primera línea de este. La línea se formatea para que tenga codificación UTF-8, pero no es necesario agregarla a la base de datos ya que solo posee información sobre los campos que vienen en el fichero.

```php
//abre el fichero
$archive = fopen($_FILES['file']['tmp_name'], "r");
 //obtiene la primera linea y lo codifica a utf8
$line = utf8_encode(fgets($archive));
```

Figura 5.1.5.2 Apertura del fichero

El sistema obtiene la segunda línea del fichero y si contiene datos, crea una variable para cada dato y los inserta en la base de datos mediante una consulta. Este proceso se realiza para todas las filas de datos del fichero.

```php
while (($data = fgetcsv($archive)) !== false) {
    $curso = utf8_encode($data[0]);
    $materia = utf8_encode($data[1]);
    $unidad = utf8_encode($data[2]);
    $profesor = utf8_encode($data[3]);

    $query = sprintf('INSERT INTO materia (nombre, cod_sist_calificacion, curso, unidad, profesor) '
            . 'VALUES ("%s", "%s", "%s", "%s", "%s")', $materia, NULL, $curso, $unidad, $profesor);
    $res = mysqli_query($connect, $query) or die(mysqli_connect_error() . "error al importar las asignaturas impartidas");
}
```

Figura 5.1.5.3 Introducción de los datos en la base de datos

Para terminar, el sistema cierra el fichero y vuelve a la página principal de Gestión de asignaturas y se muestra un mensaje de que se ha cargado el fichero correctamente.

```
fclose($archive);
$urlr = $url."gasignatura.php?ok=1&id_login=" . $_GET['id_login'];
header('Location: ' . $urlr);
exit();
```

Figura 5.1.5.4 Redirección a Gestión de asignaturas

5.1.6 Importador de unidades

La finalidad de esta sección de código contenida en *uploadUnidad.php*, del que se hará uso en la sección de Gestión de unidades (apartado 5.2.1), es comprobar que el formato del fichero es el adecuado, recoger todos los datos que vienen en él y añadirlos a la entidad *unidad* de la base de datos. A continuación se detallará paso por paso cada parte de este código.

Lo primero que hace es cargar el archivo *BDConnection.php* y asegurarse de que no hay errores al cargarlo. Si no hay errores, se asegura de que el formato del fichero es CSV. En el caso de que no lo sea, vuelve a la página principal de Gestión de asignaturas y se muestra un mensaje de error de que el fichero no tiene el formato adecuado.

```
<?php require('../database/BDConnection.php'); ?>

<?php
//comprueba que el fichero se cargó correctamente
if (isset($_FILES['file']) && $_FILES['file']['error'] === UPLOAD_ERR_OK) {
    $fileName = $_FILES['file']['name'];
    $fileComps = explode(".", $fileName); //partes del nombre del fichero
    $ext = strtolower(end($fileComps)); //extension del archivo
    if (strcmp($ext, "csv") != 0) { //comprueba que el formato del archivo es el adecuado
        //mensaje de error y vuelta ala página de carga
        fclose($archive);
        $urlr = $url."gunidad.php?ft=0&id_login=" . $_GET['id_login'];
        header('Location: ' . $urlr);
        exit();
    }
}
```

Figura 5.1.6.1 Comprobación de errores del fichero

Si no hubo ningún error ni en la carga ni en el formato del archivo, se procede a abrirlo con permisos de lectura y se obtiene la primera línea de este. La línea se formatea para que tenga codificación UTF-8, pero no es necesario agregarla a la base de datos ya que solo posee información sobre los campos que vienen en el fichero.

```
//abre el fichero
$archive = fopen($_FILES['file']['tmp_name'], "r");
 //obtiene la primera linea y lo codifica a utf8
$line = utf8_encode(fgets($archive));
```

Figura 5.1.6.2 Apertura del fichero

El sistema obtiene la segunda línea del fichero y si contiene datos, crea una variable para cada dato y los inserta en la base de datos mediante una consulta. Este proceso se realiza para todas las filas de datos del fichero.

```
while (($data = fgetcsv($archive)) !== false) {

    $unidad = utf8_encode($data[0]);
    $tipo = utf8_encode($data[1]);
    $cap = utf8_encode($data[2]);
    $numA = utf8_encode($data[3]);
    $tutor = utf8_encode($data[4]);
    $sede = utf8_encode($data[5]);
    $turno = utf8_encode($data[6]);
    $curso = utf8_encode($data[7]);

    $query = sprintf('INSERT INTO unidad (unidad, tipo, capacidad_prevista, '
            . 'num_alumnos, tutor, sede, turno_tarde, curso) VALUES ("%s", '
            . '"%s", "%s", "%s", "%s", "%s", "%s", "%s")',
            $unidad, $tipo, $cap, $numA, $tutor, $sede, $turno, $curso);
    $res = mysqli_query($connect, $query) or die(mysqli_connect_error());
}
```

Figura 5.1.6.3 Introducción de los datos en la base de datos

Para terminar, el sistema cierra el fichero y vuelve a la página principal de Gestión de asignaturas y se muestra un mensaje de que se ha cargado el fichero correctamente.

```
fclose($archive);//cierra el archivo
$urlr = $url."gunidad.php?ok=1&id_login=" . $_GET['id_login'];
header('Location: ' . $urlr);
```

Figura 5.1.6.4 Redirección a Gestión de unidades

5.1.7 Roles de usuario

Ya que esta aplicación la usarán tanto los profesores como el equipo directivo, se dispondrá de varios roles de usuario que acotarán las acciones que puede realizar cada persona que inicie sesión. Estos roles de usuario son *admin* (administrador), *profesor* y *tutor*, todos ellos se encuentran introducidos en la tabla *roles* de la base de datos e irán enlazados a cada registro de la tabla *login* del usuario. A continuación se describirá hacia quién está orientado cada rol y la funcionalidad que tienen disponible.

- **Admin:** también llamado rol de administrador, un usuario con este rol puede ejecutar todas opciones posibles del sistema, ya sea añadir, editar, consultar o eliminar datos, así como importar los ficheros CSV de Séneca o crear informes de alumnado y descargar el correspondiente PDF. Por tanto, este rol está orientado a los usuarios de la directiva del centro educativo, que además de que puedan dar clase, son los encargados de gestionar todos los datos de los docentes y los alumnos.

- **Profesor:** un usuario con este rol tendrá la opción de consultar los datos de los alumnos, asignaturas, unidades o del resto de profesores, además de poder editar informes del alumnado rellenando o editando su apartado dentro del informe y editar sus propios datos ya que puede darse el caso de que se vaya de baja y tenga que cambiar su estado y rellenar las tareas que deja al profesor que lo sustituya. Este rol lo tendrán los usuarios de la aplicación que son docentes.
- **Tutor:** un usuario con este rol podrá realizar las mismas funciones que el usuario con rol de profesor, además de poder crear informes de tutoría para los alumnos de cuya unidad él es tutor y descargar en PDF esos informes ya rellenos por todos los profesores. Los tutores tendrán en su correspondiente registro de la tabla login tanto el rol de tutor como el rol de profesor. Dicho esto, los usuarios con este rol serán los tutores de las unidades de cada curso.

5.2 Segunda iteración

Esta iteración es relativa al desarrollo de CRUDs de datos del profesorado, unidades y asignaturas, así como las páginas principales de gestión de cada uno de estos apartados. Las secciones están ordenadas según su necesidad y uso, es decir, no puedes añadir asignaturas si no se han añadido antes las unidades y los profesores. El motivo de esto es que cada asignatura tiene su unidad y cada profesor imparte una asignatura.

Ha de tenerse en cuenta que, tras la introducción de los roles de usuario en el apartado 5.1.7, no todos los botones que aparecen en las páginas estarán disponibles para ser usados, esto dependerá del rol que tenga el usuario que inicie sesión. Pero para dar una visión completa y poder detallar cada una de las funcionalidades se mostrará el sistema como si hubiera iniciado sesión un usuario con el rol de *admin*.

5.2.1 Gestión de unidades

La página principal de la sección de Gestión de unidades es la siguiente:

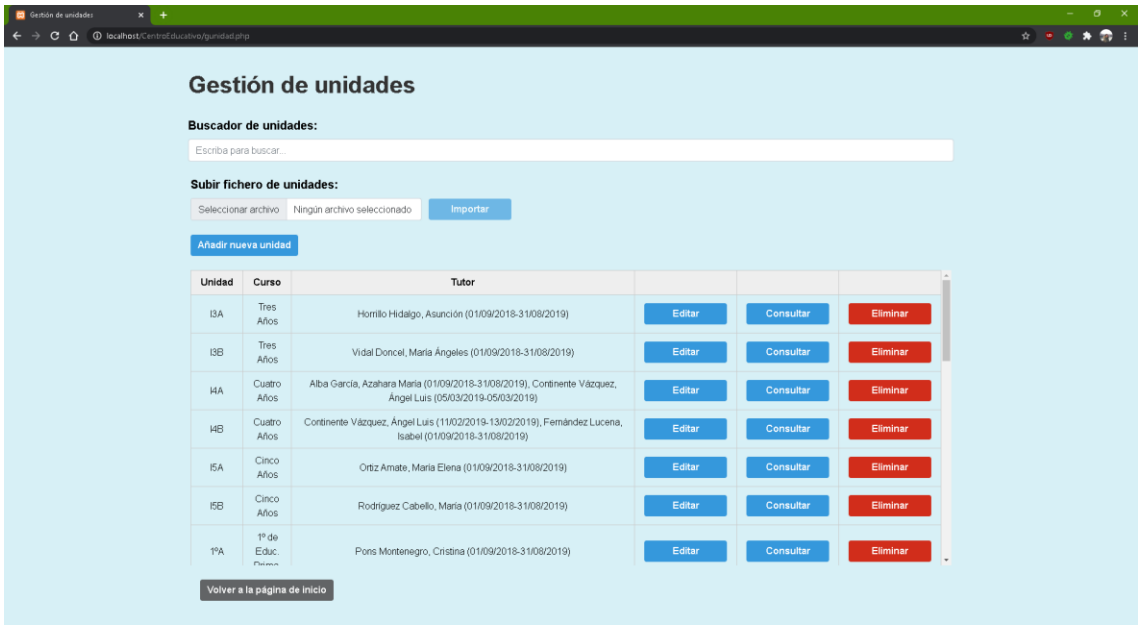

Figura 5.2.1.1 Página principal de Unidades

En esta página, lo primero que encontramos es el buscador en el cual se puede buscar por curso, por tutor o por unidad, es decir, si se quiere buscar todas las unidades de infantil solo es necesario poner en el buscador la palabra *Años* y en la tabla aparecería la lista. Este buscador, va actualizando la lista de la tabla conforme el usuario va escribiendo, por lo que no es necesario pulsar ningún botón para buscar el texto introducido.

```php
<?php
$ql = mysqli_query($connect, "SELECT * FROM login WHERE id_login='" . $_GET['id_login'] . "'");
$l = mysqli_fetch_assoc($ql);
?>
<div class="signup-form">
    <!-- buscar una unidad en la base de datos -->
    <div class="container">
        <label for="buscador" class="form-label">Buscador de unidades: </label>
        <input class="form-control" id="buscador" type="text" autocomplete="off" placeholder="Escriba para buscar..." /><br>
    </div>
```

Figura 5.2.1.2 Código HTML del buscador

La función que realiza la acción de buscar lo que hace es comprobar si alguna fila tiene algún dato que coincida con lo buscado, en cuyo caso la muestra, si no coincide la esconde.

```javascript
$(document).ready(function () {
    $("#buscador").keyup(function () {
        _this = this;
        // muestra solo las filas que coinciden con lo buscado
        $.each($("#busqueda tbody tr"), function () {
            if ($(this).text().toLowerCase().indexOf($(_this).val().toLowerCase()) === -1)
                $(this).hide();
            else
                $(this).show();
        });
    });
});
```

Figura 5.2.1.3 Función JavaScript del buscador

Lo siguiente que encontramos en la página es el importador de archivos, en este caso de unidades, el usuario debe seleccionar una archivo pulsando el botón *Seleccionar archivo*, a continuación se abrirá el explorador de archivos del ordenador, el usuario escogerá el archivo que quiere importar y pulsará el botón *Importar*. Hasta que no se seleccione un archivo, el botón *Importar* estará deshabilitado, es decir, será visible pero no podrá pulsarse. La acción que realiza el sistema tras la pulsación del botón se ha detallado en el apartado 5.1.6.

Esta opción solo aparecerá en caso de que el usuario que ha iniciado sesión tenga el rol de *admin*, ya que es el único rol de usuario que permitirá añadir, modificar o eliminar datos.

No hay ningún tipo de control sobre el número de veces que se puede subir el archivo, es decir, el sistema no controla que se suba la misma unidad dos veces, ya que puede haber datos distintos como por ejemplo que haya dos 6ºA pero en una se imparte en turno de mañana y otra en el turno de tarde. En caso de que se haya subido dos veces la misma unidad, el usuario puede buscarla y borrar la que quiera.

```
<!-- si el usuario que ha iniciado sesión, tiene el rol de admin -->
<?php if (strpos($l['rol'], "1") !== false) { ?>
    <!-- Importar un conjunto de unidades a traves de un csv -->
    <div class="input-group row btn-toolbar">
        <form id="form" method="POST" enctype="multipart/form-data" action="File import/uploadUnidad.php&id_login=<?php echo $_GET['id_login']; ?>">
            <label for="file" class="form-label">Subir fichero de unidades:</label><br>
            <div class="btn-group me-2 ">
                <input type="file" name="file" class="form-control"/>
            </div>
            <div class="btn-group me-2">
                <button id="importar" class="btn btn-primary" type="submit" disabled>Importar</button>
            </div>
        </form>
    </div><br>
```

Figura 5.2.1.4 Código HTML del importador

Si hubo errores al importar el archivo o todo se importó correctamente, el sistema mostrará un mensaje notificando al usuario lo que ha ocurrido. Esta acción se repetirá al añadir o editar los datos. Si la notificación es de color rojo, es que hubo errores, si la notificación es de color verde es que la acción se llevó a cabo con éxito.

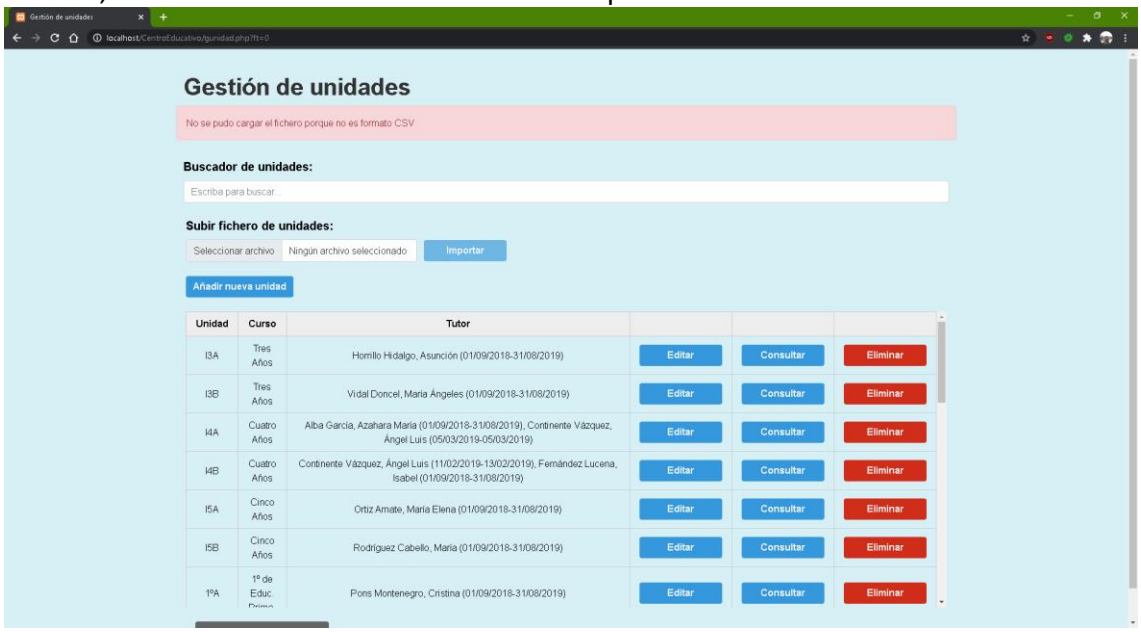

Figura 5.2.1.5 Página principal con alerta

Según el parámetro que se reciba por URL y el valor del parámetro, el sistema mostrará una alerta u otra. Si es *ft* el parámetro que recibe con valor 0 significa que hubo un error en el formato del archivo, si recibe el parámetro *ok* con valor 1 significa que el fichero se ha importado correctamente, si el parámetro es *add* con valor 1 significa que se han añadido los datos correctamente y si es *ed* con valor 1 es que se han editado los datos correctamente.

```php
<?php //mensajes de alerta para la subida de archivos y datos
if (isset($_GET['ft']) && $_GET['ft'] == 0) { //error formato archivo
    echo '<div class="container alert alert-danger" role="alert">No se pudo cargar el fichero porque no es formato CSV</div>';
}
if (isset($_GET['ok']) && $_GET['ok'] == 1) { //importacion correcta
    echo '<div class="container alert alert-success" role="alert">Se han importado los datos correctamente</div>';
}
if (isset($_GET['add']) && $_GET['add'] == 1) { //datos añadidos correctos
    echo '<div class="container alert alert-success" role="alert">Se ha añadido los datos correctamente</div>';
}
if (isset($_GET['ed']) && $_GET['ed'] == 1) { //datos editados correctos
    echo '<div class="container alert alert-success" role="alert">Se ha editado los datos correctamente</div>';
}
?>
```

Figura 5.2.1.6 Código PHP de alertas

La siguiente opción que encontramos en la página principal es la de añadir unidad nueva, es un botón que lleva al usuario a un formulario llamado *addUnidad.php*..

```html
<!-- añadir una unidad a la base de datos -->
<div class="col">
    <button name="nuevo" role="link" class="btn btn-primary" onclick="window.location = '<?php echo $url; ?>addUnidad.php?id_login=<?php echo $_GET['id_login']; ?>'">Añadir nueva unidad</button>
</div><br>
```

Figura 5.2.1.7 Código HTML botón añadir unidad

En este formulario, el usuario tendrá que agregar los datos de la nueva unidad que desea añadir y pulsar el botón *Añadir*. Si en la etiqueta del campo hay un asterisco (*) significa que el campo es obligatorio de rellenar, es decir, no se podrán guardar los datos a no ser que todos los campos con asterisco estén rellenos. Al igual que el importador, el usuario con rol de *admin*, será el único que podrá realizar esta acción.

```
<!-- formulario para rellenar los datos de la unidad -->
<div class="container signup-form">
    <form class=" row g-3" action="lib/saveAddUnidad.php" method="POST" enctype="multipart/form-data">
        <div class="col-md-4">
            <label for="unidad" class="form-label">Introduzca la unidad:* </label>
            <input name="unidad" class="form-control" type="text" required />
        </div>
        <div class="col-md-4">
            <label for="tipo" class="form-label">Introduzca el tipo de unidad:* </label>
            <input name="tipo" class="form-control" type="text" required/>
        </div>
        <div class="col-md-4">
            <label for="capacidad_prevista" class="form-label">Introduzca la capacidad prevista:* </label>
            <input name="capacidad_prevista" class="form-control" type="text" required/>
        </div>
        <div class="col-md-3">
            <label for="num_alumnos" class="form-label">Introduzca el n&uacute;mero de alumnos:* </label>
            <input name="num_alumnos" class="form-control" type="text" required/>
        </div>
        <div class="col-md-3">
            <label for="curso" class="form-label">Introduzca el curso:* </label>
            <input name="curso" class="form-control" type="text" required/>
        </div>
        <div class="col-md-3">
            <label for="sede" class="form-label">Introduzca la sede: </label>
            <input name="sede" class="form-control" type="text" />
        </div>
```

Figura 5.2.1.8 Código HTML formulario añadir unidad (1)

El campo para seleccionar si la unidad tiene turno de tarde es un checkbox con los argumentos *sí* o *no*, por defecto estará seleccionado el *no* porque al ser un colegio, lo normal es que todas las clases sean impartidas en el turno de mañana.

Además, en el apartado para introducir los tutores, se dispondrá de un cuadro de texto para que además del nombre del tutor, se pueda introducir la fecha durante la cual el profesor introducido ha sido tutor de esa unidad. Esto es necesario en el caso de que el tutor pida la baja y venga un profesor nuevo a sustituirlo, este nuevo profesor será añadido con la fecha en la que se unió y se podrá cambiar la fecha en la que el tutor "oficial" se dio de baja. Esta acción se podrá realizar en la edición de la unidad.

```
<div class="col-md-3">
    <label class="form-label">¿Es turno de tarde?:* </label><br>
    <div class="form-check form-check-inline">
        <input class="form-check-input" type="radio" name="turno_tarde" id="turnoS" value="Si">
        <label class="form-check-label" for="turnoS">Si</label>
    </div>
    <div class="form-check form-check-inline">
        <input class="form-check-input" type="radio" name="turno_tarde" id="turnoN" value="No" checked>
        <label class="form-check-label" for="turnoN">No</label>
    </div>
</div>
<div class="col-md-12">
    <label for="tutor" class="form-label">Introduzca los tutores:* </label>
    <textarea name="tutor" class="form-control" type="email" required></textarea>
</div>
<!-- boton submit formulario -->
<div class="d-flex justify-content-end bd-highlight mb-3">
    <button type="submit" class="btn btn-primary">A&ntilde;adir</button>
</div>
```

Figura 5.2.1.9 Código HTML formulario añadir unidad (2)

El formulario también posee un botón llamado Volver a Gestión de unidades el cual redirecciona a la página *gunidad.php*.

```
<!-- boton para volver a la pagina principal de gestión de unidades-->
<div class="d-flex justify-content-start bd-highlight mb-3">
    <button role="link" class="btn btn-secondary" onclick="window.location = '<?php echo $url;?>gunidad.php?id_login=<?php echo $_GET['id_login']; ?>'">Volver a Gesti&oacute;n de unidades</button>
</div>
```

Figura 5.2.1.10 Código HTML botón volver

Tras pulsar el botón *Añadir* el sistema llama al archivo *saveAddUnidad.php* que es el encargado de añadir a la base de datos. El sistema realiza la consulta con los datos obtenidos mediante el método POST y la ejecuta, por último redirige a *gunidad.php* y manda por URL el parámetro *add* para que muestre la alerta correcta.

```
$query = sprintf('INSERT INTO unidad (unidad, tipo, capacidad_prevista, num_alumnos, tutor, sede, turno_tarde, curso) VALUES ("%s", "%s", "%s", "%s", "%s", "%s", "%s", "%s")',
        $_POST['unidad'], $_POST['tipo'], $_POST['capacidad_prevista'], $_POST['num_alumnos'], $_POST['tutor'], $_POST['sede'], $_POST['turno_tarde'], $_POST['curso']); //consulta
echo $res = mysqli_query($connect, $query) or die(mysqli_connect_error() . 'error al insertar la asignatura'); //ejecuta la consulta

$urlr = $url."gunidad.php?add=1"."&id_login=" . $_GET['id_login'];; //url a la que redireccionar
header('Location: ' . $urlr); //redirecciona a la url anterior
```

Figura 5.2.1.11 Consulta INSERT unidad

A continuación se muestra el formulario tal y como lo vería el usuario:

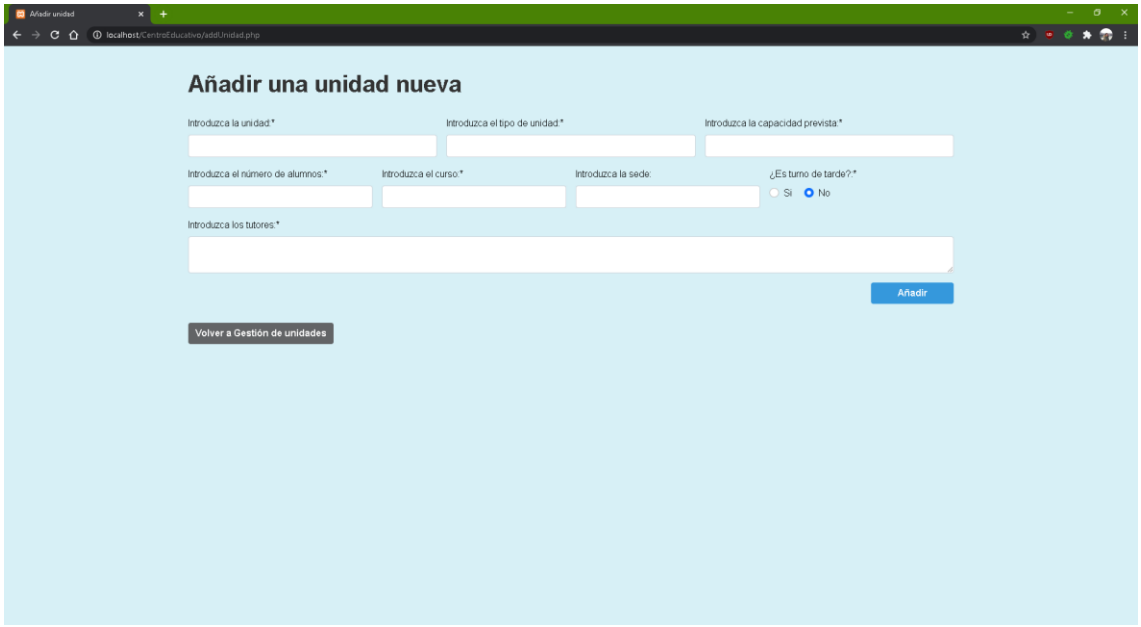

Figura 5.2.1.12 Formulario añadir unidad

Tras el botón de Añadir nueva unidad, aparece la tabla de datos. Para rellenar esta tabla, el sistema realiza una consulta SELECT que obtiene la lista de todas las unidades que se han introducido anteriormente. En caso de que la consulta no devuelva nada, la tabla solo mostrará la cabecera.

```
$luni = mysqli_query($connect, "SELECT * FROM unidad") or die(mysqli_connect_error()); //obtiene todos las unidades de la base de datos
$row = mysqli_fetch_assoc($luni);
```

Figura 5.2.1.13 Consulta SELECT unidades

Una vez tenemos la lista, por cada unidad se mostrará una fila con el nombre de la unidad, el curso al que pertenece y los tutores, además de tres opciones que se detallarán más adelante.

```
<table>
    <thead>
        <tr>
            <th>Unidad</th>
            <th>Curso</th>
            <th>Tutor</th>
            <th></th>
            <th></th>
            <th></th>
        </tr>
    </thead>
    <?php while ($row != NULL) { ?>
        <tbody>
            <tr>
                <td><?php echo $row['unidad']; ?></td>
                <td><?php echo $row['curso']; ?></td>
                <td><?php echo $row['tutor']; ?></td>
```

Figura 5.2.1.14 Código HTML tabla de datos

Estas tres opciones, mencionadas anteriormente, que hacen uso de los datos son *Editar*, *Consultar* y *Eliminar*. Cada opción es un botón que, o bien te lleva a un formulario o bien realiza la acción que su propio nombre indica. Las opciones *Editar* y *Eliminar* solo se podrán realizar si el usuario que ha iniciado sesión tiene el rol de admin.

```
<?php if (strpos($l['rol'], "1") !== false) { ?>
    <td><a class="btn btn-primary" href='<?php echo $url; ?>editUnidad.php?id=<?php echo $row['id_unidad']; ?>&id_login=<?php echo $_GET['id_login']; ?>'>Editar</a></td>
<?php } else { ?>
    <td><button class="btn btn-primary" disabled>Editar</button></td>
<?php } ?>
<td><a class="btn btn-primary" href='<?php echo $url; ?>showUnidad.php?id=<?php echo $row['id_unidad']; ?>&id_login=<?php echo $_GET['id_login']; ?>'>Consultar</a></td>
<?php if (strpos($l['rol'], "1") !== false) { ?>
    <td><button class="delete btn btn-danger" id="<?= $row['id_unidad'] ?>" data-id="<?= $row['id_unidad'] ?>" >Eliminar</button></td>
<?php } else { ?>
    <td><button class="delete btn btn-danger" id="<?= $row['id_unidad'] ?>" data-id="<?= $row['id_unidad'] ?>" disabled>Eliminar</button></td>
<?php } ?>
```

Figura 5.2.1.15 Código HTML opciones de tabla

Si el usuario pulsa el botón Editar, se muestra la página *editUnidad.php* a la cual se le ha enviado la *id* de la unidad que se quiere editar a través de la URL y el sistema realizará la consulta SELECT para obtener los datos.

```
$id = $_GET['id']; //coge el id de la unidad de la url
//obtiene todos los datos de la unidad de la base de datos
$query = mysqli_query($connect, 'SELECT * FROM unidad WHERE id_unidad="' . $id . '"') or die(mysqli_connect_error() . "no coge el id");
$uni = mysqli_fetch_assoc($query);
```

Figura 5.2.1.16 Consulta SELECT de una unidad

Una vez que tenemos los datos de la unidad, se realiza un formulario el cual ya está relleno.

```
<form id="form" class=" row g-3" action="lib/saveEditUnidad.php?id=<?= $uni['id_unidad']; ?>" method="POST" enctype="multipart/form-data">
    <div class="col-md-4">
        <label for="unidad" class="form-label">Unidad:* </label>
        <input name="unidad" class="form-control" type="text" required value="<?= $uni['unidad'] ?>"/>
    </div>
    <div class="col-md-4">
        <label for="tipo" class="form-label">Tipo de unidad:* </label>
        <input name="tipo" class="form-control" type="text" required value="<?= $uni['tipo'] ?>"/>
    </div>
    <div class="col-md-4">
        <label for="capacidad_prevista" class="form-label">Capacidad prevista:* </label>
        <input name="capacidad_prevista" class="form-control" type="text" required value="<?= $uni['capacidad_prevista'] ?>"/>
    </div>
    <div class="col-md-3">
        <label for="num_alumnos" class="form-label">N&uacute;mero de alumnos:* </label>
        <input name="num_alumnos" class="form-control" type="text" required value="<?= $uni['num_alumnos'] ?>"/>
    </div>
    <div class="col-md-3">
        <label for="curso" class="form-label">Curso:* </label>
        <input name="curso" class="form-control" type="text" required value="<?= $uni['curso'] ?>"/>
    </div>
    <div class="col-md-3">
        <label for="sede" class="form-label">Sede: </label>
        <input name="sede" class="form-control" type="text" value="<?= $uni['sede'] ?>"/>
    </div>
```

Figura 5.2.1.17 Código HTML formulario editar unidad (1)

En el caso del turno de la unidad sigue siendo un checkbox en el que está marcado el turno que se guardó al importar o añadir la unidad. En cuanto a los tutores, se muestra un cuadro de texto con el tutor que tenía la unidad cuando se importó.

```
<div class="col-md-3">
    <label class="form-label">¿Es turno de tarde?:* </label><br>
    <?php if (strcmp($uni['turno_tarde'], "No") == 0) { ?>
        <div class="form-check form-check-inline">
            <input class="form-check-input" type="radio" name="turno_tarde" id="turnoS" value="Si">
            <label class="form-check-label" for="turnoS">Si</label>
        </div>
        <div class="form-check form-check-inline">
            <input class="form-check-input" type="radio" name="turno_tarde" id="turnoN" value="No" checked>
            <label class="form-check-label" for="turnoN">No</label>
        </div>
    <?php } else { ?>
        <div class="form-check form-check-inline">
            <input class="form-check-input" type="radio" name="turno_tarde" id="turnoS" value="Si" checked>
            <label class="form-check-label" for="turnoS">Si</label>
        </div>
        <div class="form-check form-check-inline">
            <input class="form-check-input" type="radio" name="turno_tarde" id="turnoN" value="No" >
            <label class="form-check-label" for="turnoN">No</label>
        </div>
    <?php } ?>
</div>
<div class="col-md-12">
    <label for="tutor" class="form-label">Tutores:* </label>
    <textarea name="tutor" class="form-control" type="email" required value="<?= $uni['tutor'] ?>"><?= $uni['tutor'] ?></textarea>
</div>
<div class="d-flex justify-content-end bd-highlight mb-3">
    <button id="editar" type="submit" class="btn btn-primary"disabled>Editar</button>
</div>
```

Figura 5.2.1.18 Código HTML formulario editar unidad (2)

En el momento en el que el usuario modifique algún dato, el botón llamado *Editar* que se encuentra al final del formulario se habilitará. La función que habilita el botón, es un función JavaScript que detecta si hay algún cambio en los input del formulario y activa el botón.

```
<script>
    $(document).ready(function () {
        $('#form').on('input change', function () {
            $('#editar').attr('disabled', false);
        });
    })
</script>
```

Figura 5.2.1.19 Función JavaScript que habilita el botón

Por último, el usuario pulsará el botón y el sistema llamará al archivo *saveEditUnidad.php* que realizará la consulta UPDATE con los parámetros que se obtuvieron utilizando POST y actualizará los datos de la unidad. Para saber qué unidad tenemos que actualizar, se pasará por URL la *id* de la unidad.

Una vez actualizados los datos, el sistema volverá a la página principal de la gestión de unidades y se mostrará un mensaje diciendo que se han editado los datos correctamente.

```
$id = $_GET['id']; //coge el id de la url
$query = "UPDATE unidad SET unidad='" . $_POST['unidad'] . "', tipo='" . $_POST['tipo'] . "', capacidad_prevista='" . $_POST['capacidad_prevista']
        . "', num_alumnos='" . $_POST['num_alumnos'] . "', " . "tutor='" . $_POST['tutor'] . "', sede='" . $_POST['sede'] . "', turno_tarde='"
        . $_POST['turno_tarde'] . "',curso='" . $_POST['curso'] . "'  WHERE id_unidad='" . $id . "'"; // consulta
echo $res = mysqli_query($connect, $query) or die(mysqli_connect_error() . 'error al editar el profesor'); //ejecuta la consulta

$urlr = $url."gunidad.php?ed=1"."&id_login=" . $_GET['id_login'];; //url a la que redirreciona
header('Location: ' . $urlr); //redirecciona a gunidad
```

Figura 5.2.1.20 Consulta UPDATE unidad

Como el resto de formularios de esta sección, también posee un botón llamado Volver a Gestión de unidades el cual redirecciona a la página *gunidad.php*.

A continuación se mostrará un ejemplo del formulario de edición de la unidad:

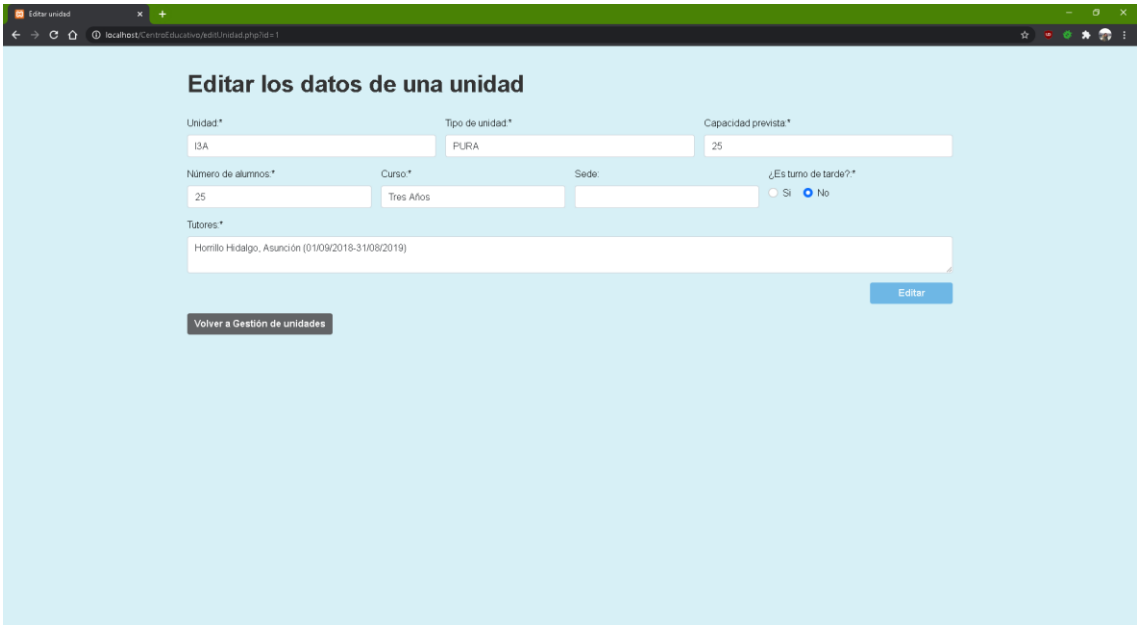

Figura 5.2.1.21 Formulario editar unidad

Es la segunda opción disponible para cada fila de datos de la tabla. Cuando el usuario pulsa el botón *Consultar*, el sistema muestra la página *showUnidad.php* la cual es un formulario que muestra los datos de esa unidad en concreto. Para ello se envía por URL la id de la unidad que se desea consultar y el sistema realiza una consulta SELECT para obtener los datos de la base de datos.

```php
$id = $_GET['id']; //coge el id de la unidad de la url
//obtiene todos los datos de la unidad de la base de datos
$query = mysqli_query($connect, 'SELECT * FROM unidad WHERE id_unidad="' . $id . '"') or die(mysqli_connect_error() . "no coge el id");
$uni = mysqli_fetch_assoc($query);
```

Figura 5.2.1.22 Consulta SELECT unidad

Una vez se han obtenido los datos de la base de datos, el sistema muestra un formulario con los datos de la unidad. Estos datos no se pueden editar, solo se muestran a modo de consulta para el usuario.

```php
<form id="form" class=" row g-3" enctype="multipart/form-data">
    <div class="col-md-4">
        <label for="unidad" class="form-label">Unidad: </label>
        <input name="unidad" class="form-control" type="text" required readonly disabled value="<?= $uni['unidad'] ?>"/>
    </div>
    <div class="col-md-4">
        <label for="tipo" class="form-label">Tipo de unidad: </label>
        <input name="tipo" class="form-control" type="text" required readonly disabled value="<?= $uni['tipo'] ?>"/>
    </div>
    <div class="col-md-4">
        <label for="capacidad_prevista" class="form-label">Capacidad prevista: </label>
        <input name="capacidad_prevista" class="form-control" type="text" required readonly disabled value="<?= $uni['capacidad_prevista'] ?>"/>
    </div>
    <div class="col-md-3">
        <label for="num_alumnos" class="form-label">N&uacute;mero de alumnos: </label>
        <input name="num_alumnos" class="form-control" type="text" required readonly disabled value="<?= $uni['num_alumnos'] ?>"/>
    </div>
    <div class="col-md-3">
        <label for="curso" class="form-label">Curso: </label>
        <input name="curso" class="form-control" type="text" required readonly disabled value="<?= $uni['curso'] ?>"/>
    </div>
    <div class="col-md-3">
        <label for="sede" class="form-label">Sede: </label>
        <input name="sede" class="form-control" type="text" readonly disabled value="<?= $uni['sede'] ?>"/>
    </div>
    <div class="col-md-3">
        <label class="form-label">¿Es turno de tarde?: </label><br>
        <input class="form-control" type="text " name="turno_tarde" readonly disabled value="<?= $uni['turno_tarde'] ?>">
    </div>
    <div class="col-md-12">
        <label for="tutor" class="form-label">Tutores: </label>
        <textarea name="tutor" class="form-control" type="email" required readonly disabled value="<?= $uni['tutor'] ?>"><?= $uni['tutor'] ?></textarea>
    </div>
</form><br>
```

Figura 5.2.1.23 Código HTML formulario consultar unidad

Como el resto de formularios de esta sección, también posee un botón llamado Volver a Gestión de unidades el cual redirecciona a la página *gunidad.php*.

```php
<!-- boton para volver a la pagina principal de gestion del profesorado -->
<div class="d-flex justify-content-start bd-highlight mb-3">
    <button role="link" class="btn btn-secondary" onclick="window.location = 'http://localhost/CentroEducativo/gunidad.php'">Volver a Gesti&oacute;n de unidades</button>
</div>
```

Figura 5.2.1.124 Código HTML botón volver atrás

A continuación se mostrará un ejemplo del formulario de consulta de la unidad:

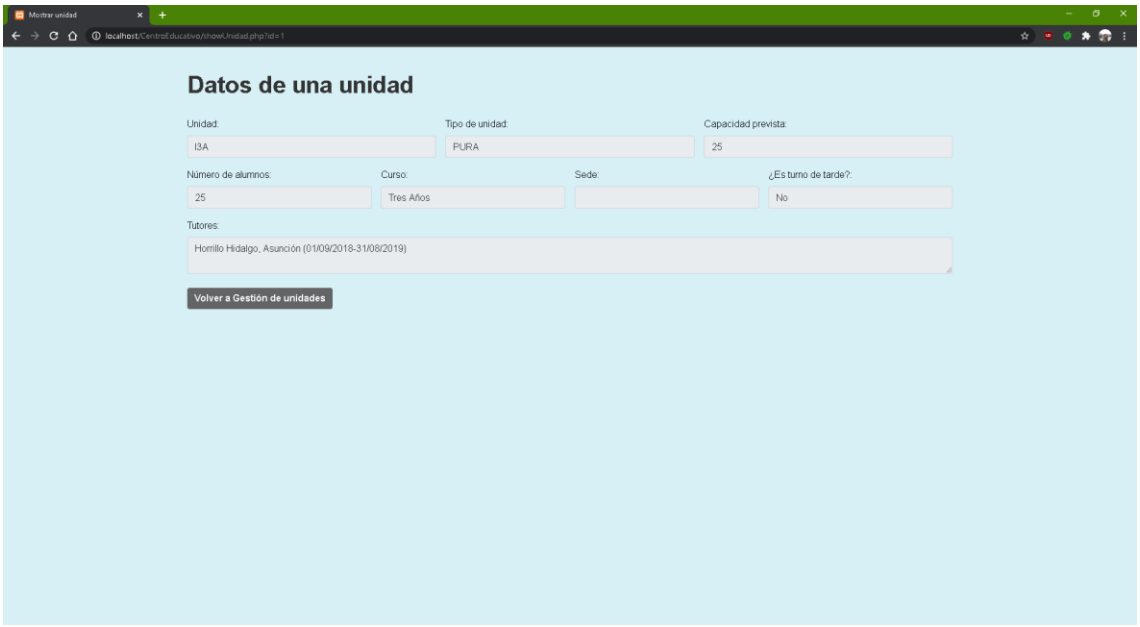

Figura 5.2.1.25 Formulario consultar unidad

La última acción a realizar sobre cada unidad es la de eliminar. El usuario pulsará el botón *Eliminar* y el sistema ejecutará el siguiente código JavaScript que lo que hace es mostrar una mensaje preguntando si el usuario realmente desea eliminar el registro. En caso de que el usuario cierre el mensaje o haga click en la opción *Cancel*, se cerrará el mensaje y el sistema no eliminará nada. Para que el sistema muestre el mensaje, se usará bootbox.

```
// funcion para borrar filas de la tabla
$(document).ready(function () {
    $('.delete').click(function () {
        var el = this;
        //coge la id del elemento a borrar
        var deleteid = $(this).data('id');
        // muestra caja de confirmación
        bootbox.confirm("¿Realmente quiere eliminar?", function (result) {
            if (result) {
                //borra el elemento y actualiza la tabla con ajax
                $.ajax({
                    url: 'lib/deleteUnidad.php',
                    type: 'POST',
                    data: {id: deleteid},
                    success: function (response) {
```

Figura 5.2.1.26 Código JavaScript eliminar unidad (1)

En caso de que el usuario haga click en la opción *Ok*, el sistema ejecutará el archivo *deleteUnidad.php*. En este archivo, el sistema recoge la id de la unidad que se quiere eliminar, se ejecuta la consulta DELETE y muestra el número 1 en caso de que se haya podido eliminar. Si no se pudo eliminar, el sistema mostrará el número 0.

```php
if (isset($_POST['id'])) { //si el id existe y no es null
    $cod = $_POST['id'];//obtiene el id del formulario

    $query = "DELETE FROM unidad WHERE id_unidad='" . $cod . "'"; //consulta
    $res = mysqli_query($connect, $query) or die(mysqli_connect_error()
            . 'error al eliminar la unidad'); //ejecuta la consulta
    echo 1;
    exit;
} else {
    echo 0;
    exit;
}
```

Figura 5.2.1.27 Código PHP eliminar unidad

Si el sistema recibe el número 1 como respuesta, la fila que se ha eliminado de la base de datos se mostrará en rojo y desaparecerá de la tabla usando AJAX. En caso de que la respuesta haya sido 0, se mostrará un mensaje diciendo que no se pudo eliminar el registro.

```javascript
success: function (response) {
    //borra el elemento
    if (response == 1) {
        $(el).closest('tr').css('background', 'tomato');
        $(el).closest('tr').fadeOut(800, function () {
            $(this).remove();
        });
    } else { // si no se ha podido eleminar muestra el mensaje
        bootbox.alert('No se ha eliminado.');
    }
}
```

Figura 5.2.1.28 Código JavaScript eliminar unidad (2)

Para terminar este apartado, justo debajo de la tabla hay un botón para volver a la página de inicio, en el caso de que el usuario quiera acceder a otro apartado de gestión.

```php
<!-- boton para volver a home.php -->
<div class="container">
    <button role="link" class="btn btn-secondary" onclick="window.location = '<?php echo $url; ?>home.php?id_login=<?php echo $_GET['id_login']; ?>'">Volver a la página de inicio</button>
</div>
```

Figura 5.2.1.29 Código HTML volver a inicio

5.2.2 Gestión del profesorado

La página principal de la sección de Gestión del profesorado es la siguiente:

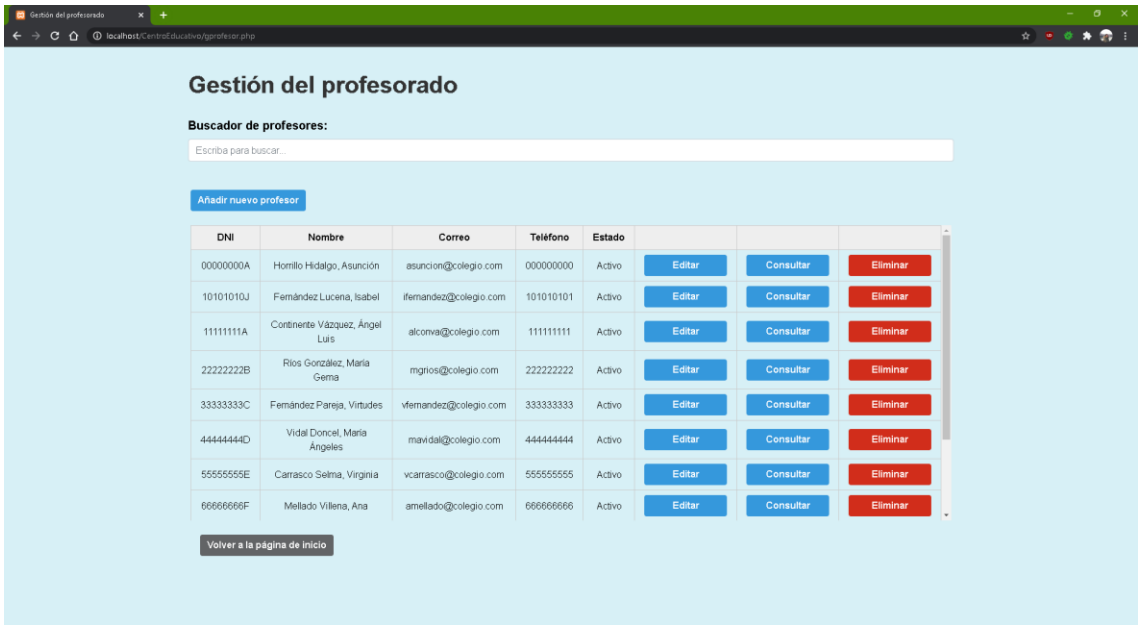

Figura 5.2.2.1 Página principal de profesorado

En esta página, lo primero que encontramos es el buscador en el cual se puede buscar por DNI, nombre, correo, teléfono o estado del profesor, es decir, si se quiere buscar todas los profesores que estén ausentes solo es necesario poner en el buscador la palabra *ausente* y en la tabla aparecería la lista. Este buscador, va actualizando la lista de la tabla conforme el usuario va escribiendo, por lo que no es necesario pulsar ningún botón para buscar el texto introducido.

```
<!-- buscar un profesor en la base de datos -->
<div class="container">
    <label for="buscador" class="form-label">Buscador de profesores: </label>
    <input class="form-control" id="buscador" type="text" autocomplete="off" placeholder="Escriba para buscar..." /><br>
</div><br>
```

Figura 5.2.2.2 Código HTML del buscador

La función que realiza la acción de buscar lo que hace es comprobar si alguna fila tiene algún dato que coincida con lo buscado, en cuyo caso la muestra, si no coincide la esconde.

```
$(document).ready(function () {
    $("#buscador").keyup(function () {
        _this = this;
        // muestra solo las filas que coinciden con lo buscado
        $.each($("#busqueda tbody tr"), function () {
            if ($(this).text().toLowerCase().indexOf($(_this).val().toLowerCase()) === -1)
                $(this).hide();
            else
                $(this).show();
        });
    });
});
```

Figura 5.2.2.3 Función JavaScript del buscador

En esta ocasión, al no tener el importador solo disponemos de alertas en función de si se han añadido o editado los datos. Si el parámetro es *add* con valor 1 significa que se han añadido los datos correctamente y si es *ed* con valor 1 es que se han editado los datos correctamente. Las alertas saldrían en el mismo lugar que en el resto de secciones.

```php
<?php //mensajes de alerta para la subida de archivos y datos
if (isset($_GET['add']) && $_GET['add'] == 1) {
    echo '<div class="container alert alert-success" role="alert">Se ha añadido los datos correctamente</div>';
}
if (isset($_GET['ed']) && $_GET['ed'] == 1) {
    echo '<div class="container alert alert-success" role="alert">Se ha editado los datos correctamente</div>';
}
?>
```

Figura 5.2.2.4 Código PHP de alertas

La siguiente opción que encontramos en la página principal es la de añadir nuevo profesor, es un botón que lleva al usuario a un formulario llamado *addProfesor.php*. En este caso, todos los profesores deberán añadirse de manera manual, ya la mayoría de los profesores continúan impartiendo clase en el mismo colegio de un curso académico al siguiente. Este botón solo aparecerá si el usuario que ha iniciado sesión tiene el rol de administrador.

```php
<?php if (strpos($_SESSION['rol'], "1") !== false) { ?>
    <div class="col">
        <button name="nuevo" role="link" class="btn btn-primary" onclick="window.location = '<?php echo $url; ?>addProfesor.php?id_login=<?php echo $_GET['id_login']; ?>'">Añadir nuevo profesor</button>
    </div><br>
<?php } ?>
```

Figura 5.2.2.5 Código HTML botón añadir profesor

En este formulario, el usuario tendrá que agregar los datos del profesor nuevo que desea añadir y pulsar el botón *Añadir*. Si en la etiqueta del campo hay un asterisco (*) significa que el campo es obligatorio de rellenar, es decir, no se podrán guardar los datos a no ser que todos los campos con asterisco estén rellenos.

```php
<form class=" row g-3" action="lib/saveAddProf.php?id_login=<?php echo $_GET['id_login']; ?>" method="POST" enctype="multipart/form-data">
    <div class="col-md-6">
        <label for="dni" class="form-label">Introduzca el DNI:* </label>
        <input name="dni" class="form-control" type="text" required maxlength="9"/>
    </div>
    <div class="col-md-6">
        <label for="nombre" class="form-label">Introduzca el nombre:* </label>
        <input name="nombre" class="form-control" type="text" required/>
    </div>
    <div class="col-md-6">
        <label for="apellido1" class="form-label">Introduzca el primer apellido:* </label>
        <input name="apellido1" class="form-control" type="text" required/>
    </div>
    <div class="col-md-6">
        <label for="apellido2" class="form-label">Introduzca el segundo apellido:* </label>
        <input name="apellido2" class="form-control" type="text" required/>
    </div>
    <div class="col-md-6">
        <label for="email" class="form-label">Introduzca el correo:* </label>
        <input name="email" class="form-control" type="email" placeholder="profesor@colegio.com" required/>
    </div>
    <div class="col-md-3">
        <label for="telefono" class="form-label">Introduzca el tel&eacute;fono:* </label>
        <input name="telefono" class="form-control" type="tel" placeholder="666666666" required />
    </div>
    <div class="col-md-3">
        <label for="rol" class="form-label">Roles del usuario:* </label><br>
        <input type="checkbox" class="form-check-input" name="rol[]" value="1"> Admin</input> 
        <input type="checkbox" class="form-check-input" name="rol[]" value="2" required> Profesor</input> 
        <input type="checkbox" class="form-check-input" name="rol[]" value="3"> Tutor</input> 
    </div>
    <!-- boton submit formulario -->
    <div class="d-flex justify-content-end bd-highlight mb-3">
        <button type="submit" class="btn btn-primary">A&ntilde;adir</button>
    </div>
```

Figura 5.2.2.6 Código HTML formulario añadir profesor

El formulario también posee un botón llamado Volver a Gestión del profesorado el cual redirecciona a la página *gprofesor.php*.

```
<div class="d-flex justify-content-start bd-highlight mb-3">
    <button role="link" class="btn btn-secondary" onclick="window.location = '<?php echo $url; ?>gprofesor.php?id_login=<?php echo $_GET['id_login']; ?>'">Volver a Gesti&oacute;n del profesorado</button>
</div>
```

Figura 5.2.2.7 Código HTML botón volver

Tras pulsar el botón *Añadir* el sistema llama al archivo *saveAddProf.php* que es el encargado de añadir a la base de datos. El sistema crea las variables *found* y *id* para guardar si el profesor ya se insertó anteriormente en la base de datos y para guardar el id con el que ese usuario se registró, si es que lo hizo. A continuación, comprueba que el DNI del profesor no ha sido añadido anteriormente, ya que cada DNI es único y personal.

```
$login = $_GET['id_login'];
$lprof = mysqli_query($connect, "SELECT * FROM profesor") or die(mysqli_connect_error()); //selecciona todos los datos de la tabla profesor
$row = mysqli_fetch_assoc($lprof); //ejecuta la consulta
$found = 0; //se pone a 1 si el profesor ya se ha insertado en la bd
$id = 0; //guarda el id del login del usuario

while ($found == 0 && $row != NULL) { //comprueba si ya hay alguien insertado con ese correo
    if (strcasecmp($row['dni_profesor'], $_POST['dni']) == 0 && strcasecmp($row['email'], $_POST['email']) == 0) { //si exite en la tabla
        $found = 1; //marca found a 1
    } else { //si no existe
        $row = mysqli_fetch_assoc($lprof); //coge el siguiente profesor de la base de datos
    }
}
```

Figura 5.2.2.8 Consulta comprueba DNI profesor

Tras esto, el sistema obtendrá todos los registros de la tabla login y comprobará si algún profesor se ha registrado en el sistema con el correo que se ha introducido en sus datos personales. En caso de que se encuentre, se pondrá la variable *foundp* a 1 y se guardará en la variable *id*, el id que se le asignó al registrarse.

```
//selecciona todos los datos de la tabla login
$lUsers = mysqli_query($connect, "SELECT * FROM login") or die(mysqli_connect_error());
$p = mysqli_fetch_assoc($lUsers);
$foundp = 0; //0 si no hay ningún profesor en la base de datos con ese correo, 1 si lo hay

//comprueba si hay un profesor en login con ese correo para relacionarlo con este profesor que se va a añadir
while ($foundp == 0 && $p != NULL) {
    if (strcasecmp($p['email'], $_POST['email']) == 0) { //si en la base de datos ya esta el correo
        $foundp = 1; //pone found a 1
        $id = $p['id_login'];
    } else { //si no está el correo sigue buscando hasta que lo encuentra o no hay más donde buscar
        $p = mysqli_fetch_assoc($lUsers);
    }
}
```

Figura 5.2.2.9 Consulta comprueba email profesor

Si se ha encontrado el DNI en la base de datos, el sistema muestra una alerta y vuelve al formulario para añadir el profesor. En caso de que no se haya encontrado, se realizará la consulta.

Cada profesor que se añade nuevo, se da de alta como *Activo* automáticamente, ya que suponemos que va a estar trabajando en cuanto se añade. Además, en caso de que no se haya registrado en la aplicación, se pondrá como *id_login* el valor 0, si se ha registrado se guardará en *id_login* el valor del id que se le asignó al registrarse.

A continuación, se realiza la consulta con los datos obtenidos mediante el método POST, la ejecuta y, en caso de que el usuario ya se haya registrado, se actualizarán los roles del usuario a los elegidos en el formulario añadir. Por último redirige a *gprofesor.php* y manda por URL el parámetro *add* para que muestre la alerta correcta.

```
if ($found == 0) { //si no hay ningun profesor con ese dni, se guarda en la base de datos
    $profe = $_POST['apellido1'] . " " . $_POST['apellido2'] . ", " . $_POST['nombre'];  //une todo el nombre del profesor en una linea
    $query = sprintf('INSERT INTO profesor (dni_profesor, profesor, apellido1, apellido2, nombre, telefono, email, estado, id_login) VALUES ("%s", "%s", "%s", "%s", "%s", "%s", "%s", "%s", "%s")',
            $_POST['dni'], $profe, $_POST['apellido1'], $_POST['apellido2'], $_POST['nombre'], $_POST['telefono'], $_POST['email'], "Activo", $id); //consulta
    echo $res = mysqli_query($connect, $query) or die(mysqli_connect_error().'error al insertar el profesor'); //ejecuta la consulta

    if ($foundp == 1) { //si el usuario ya se ha registrado, se actualizan sus roles
        $rol = $_POST['rol'][0] . " " . $_POST['rol'][1] . " " . $_POST['rol'][2];
        $q = "UPDATE login SET  rol='" . $rol . "' WHERE id_login='" . $login . "'";
        echo $rp = mysqli_query($connect, $q)or die(mysqli_connect_error() . 'error al update profesor id_login');
    }
    $urlc = $url . "gprofesor.php?add=1&id_login=" . $login; //url a la que redireccionar
    header('Location: '.$urlc); //redirecciona a la url anterior
} else { //si ya hay alguien con ese dni, vuelve a la pagina de añadir profesor
    echo '<script> alert("Ya se ha añadido un profesor con este DNI o con este correo anteriormente."); history.back();</script>'; //vuelve al formulario de añadir
}
```

Figura 5.2.2.10 Consulta INSERT profesor

A continuación se muestra el formulario tal y como lo vería el usuario:

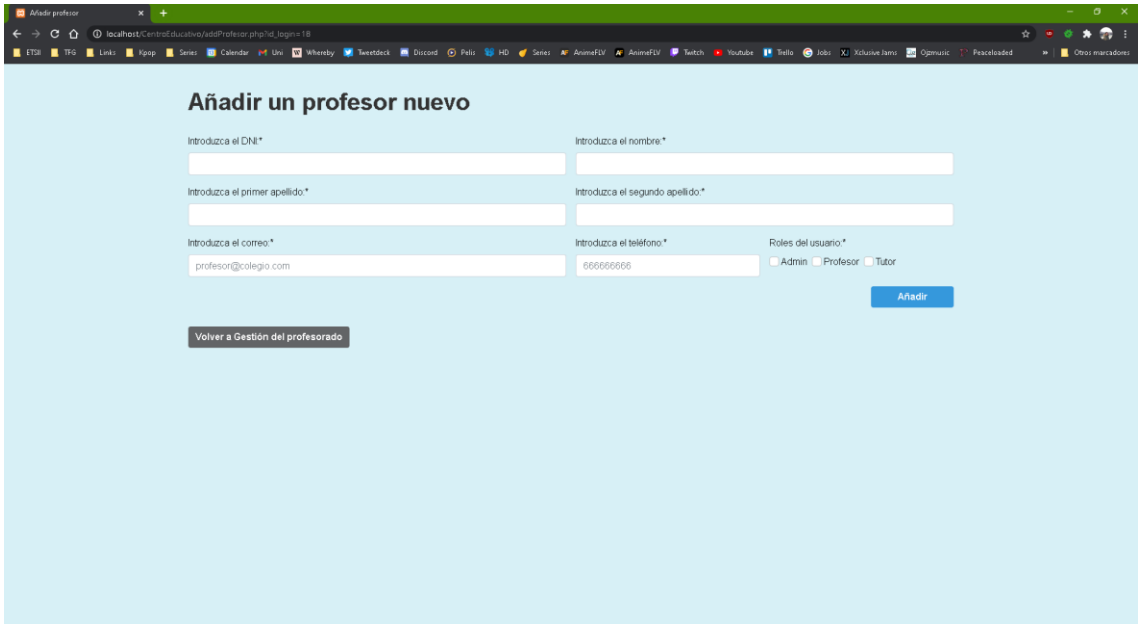

Figura 5.2.2.11 Formulario añadir profesor

Tras el botón de Añadir nuevo profesor, aparece la tabla de datos. Para rellenar esta tabla, el sistema realiza una consulta SELECT que obtiene la lista de todos los profesores que se han introducido anteriormente. En caso de que la consulta no devuelva nada, la tabla solo mostrará la cabecera.

```
//obtiene todos los profesores de la base de datos
$lprof = mysqli_query($connect, "SELECT * FROM profesor") or die(mysqli_connect_error());
$row = mysqli_fetch_assoc($lprof);
```

Figura 5.2.2.12 Consulta SELECT profesores

Una vez tenemos la lista, por cada profesor se mostrará una fila con el DNI, nombre, teléfono, correo y estado, además de tres opciones de uso de los datos.

```
<table>
    <thead>
        <tr>
            <th>DNI</th>
            <th>Nombre</th>
            <th>Correo</th>
            <th>Tel&eacute;fono</th>
            <th>Estado</th>
            <th></th>
            <th></th>
            <th></th>
        </tr>
    </thead>
    <?php while ($row != NULL) { ?>
        <tbody>
            <tr>
                <td><?php echo $row['dni_profesor']; ?></td>
                <td><?php echo $row['profesor']; ?></td>
                <td><?php echo $row['email']; ?></td>
                <td><?php echo $row['telefono']; ?></td>
                <td><?php echo $row['estado']; ?></td>
```

Figura 5.2.2.13 Código HTML tabla de datos

Estas tres opciones, mencionadas anteriormente, que hacen uso de los datos son *Editar*, *Consultar* y *Eliminar*. Cada opción es un botón que, o bien te lleva a un formulario o bien realiza la acción que su propio nombre indica.

En este caso, el único que puede eliminar los datos es un usuario con rol de administrado. Pero a lo hora de editarlos lo puede hacer tanto el administrados como el propio profesor en caso de que sean sus datos, ya que si el profesor se da de baja necesitará cambiar su estado a ausente e agregar, si es necesario, tareas para los alumnos durante los días que esté de baja.

```
<?php if (strcasecmp($l['email'], $row['email']) == 0 || strpos($l['rol'], "1") !== false) { ?>
    <td><a class="btn btn-primary" href='<?php echo $url; ?>editProfesor.php?dni=<?php echo $row['dni_profesor']; ?>&id_login=<?php echo $_GET['id_login']; ?>'>Editar</a></td>
<?php } else { ?>
    <td><button class="btn btn-primary" disabled>Editar</button></td>
<?php } ?>
<td><a class="btn btn-primary" href='<?php echo $url; ?>showProfesor.php?dni=<?php echo $row['dni_profesor']; ?>&id_login=<?php echo $_GET['id_login']; ?>'>Consultar</a></td>
<?php if (strpos($l['rol'], "1") !== false) { ?>
    <td><button class="delete btn btn-danger" id="<?= $row['dni_profesor'] ?>" data-id="<?= $row['dni_profesor'] ?>" >Eliminar</button></td>
<?php } else { ?>
    <td><button class="delete btn btn-danger" id="<?= $row['dni_profesor'] ?>" data-id="<?= $row['dni_profesor'] ?>" disabled>Eliminar</button></td>
<?php } ?>
```

Figura 5.2.2.14 Código HTML opciones de tabla

Si el usuario pulsa el botón Editar, se muestra la página *editProfesor.php* a la cual se le ha enviado el *dni* del profesor cuyos datos se quiere editar a través de la URL y el sistema realizará la consulta SELECT para obtener los datos tanto de la tabla profesor como de la tabla login de la base de datos.

```
$dni_p = $_GET['dni']; //coge el dni del profesor de la url
//obtiene todos los datos del profesor de la base de datos
$query = mysqli_query($connect, 'SELECT * FROM profesor WHERE dni_profesor="' . $dni_p . '"') or die(mysqli_connect_error() . "no coge el dni");
$prof = mysqli_fetch_assoc($query);

$lUsers = mysqli_query($connect, "SELECT * FROM login WHERE id_login ='" . $prof['id_login'] . "'") or die(mysqli_connect_error());
$p = mysqli_fetch_assoc($lUsers);
```

Figura 5.2.2.15 Consulta SELECT de un profesor

Una vez que tenemos los datos del profesor, se realiza un formulario el cual ya está relleno.

```
<form id="form" class=" row g-3" action="lib/saveEditProf.php?dni=<?= $prof['dni_profesor']; ?>" method="POST" enctype="multipart/form-data">
    <div class="col-md-6">
        <label for="dni" class="form-label">DNI: </label>
        <input class="form-control" name="dni" type="text" readonly disabled required maxlength="9" value="<?= $prof['dni_profesor']; ?>"/>
    </div>
    <div class="col-md-6">
        <label for="nombre" class="form-label">Nombre:* </label>
        <input class="form-control" name="nombre" type="text" required value="<?= $prof['nombre']; ?>"/>
    </div>
    <div class="col-md-6">
        <label for="apellido1" class="form-label">Primer apellido:* </label>
        <input class="form-control" name="apellido1" type="text" required value="<?= $prof['apellido1']; ?>"/>
    </div>
    <div class="col-md-6">
        <label for="apellido2" class="form-label">Segundo apellido:* </label>
        <input class="form-control" name="apellido2" type="text" required value="<?= $prof['apellido2']; ?>"/>
    </div>
    <div class="col-md-6">
        <label for="email" class="form-label">Correo: </label>
        <input class="form-control" name="email" type="email" readonly disabled placeholder="profesor@colegio.com" value="<?= $prof['email']; ?>"/>
    </div>
    <div class="col-md-6">
        <label for="telefono" class="form-label">Tel&eacute;fono:*</label>
        <input class="form-control" name="telefono" type="tel" placeholder="666666666" required value="<?= $prof['telefono']; ?>"/>
    </div>
```

Figura 5.2.2.16 Código HTML formulario editar profesor (1)

En cuanto a los roles que tiene asignados el usuario, se mostrarán unos checkbox en los que vendrán seleccionados los roles actuales del profesor. El único rol que obligatoriamente debe estar seleccionado es el de *profesor*, ya que se supondrá que todos los datos introducidos en esta sección son profesores.

```
<?php if (mysqli_num_rows($lUsers) > 0) { ?>
    <div class="col-md-3">
        <label for="rol" class="form-label">Roles del usuario:* </label><br>
        <?php if (strpos($p['rol'], "1") !== false) { ?>
        <input type="checkbox" class="form-check-input" name="rol[]" value="1" checked> Admin</input> 
        <?php } else { ?>
            <input type="checkbox" class="form-check-input" name="rol[]" value="1"> Admin</input> 
        <?php } ?>
        <?php if (strpos($p['rol'], "2") !== false) { ?>
            <input type="checkbox" class="form-check-input" name="rol[]" value="2" required checked> Profesor</input> 
        <?php } else { ?>
            <input type="checkbox" class="form-check-input" name="rol[]" value="2" required> Profesor</input> 
        <?php } ?>
        <?php if (strpos($p['rol'], "3") !== false) { ?>
            <input type="checkbox" class="form-check-input" name="rol[]" value="3" checked> Tutor</input> 
        <?php } else { ?>
            <input type="checkbox" class="form-check-input" name="rol[]" value="3"> Tutor</input> 
        <?php } ?>
    </div>
<?php } ?>
```

Figura 5.2.2.17 Código HTML formulario editar profesor (2)

En el caso del estado del profesor, si es *activo* no se mostrará un cuadro de texto con las tareas que debe realizar el profesor que lo sustituya, este sería el estado en el que aparecería un profesor que se acaba de introducir en la base de datos.

```
<!-- select para cambiar el estado del profesor, si se cambia a ausente aparece un textarea para rellenar con las tareas que le va a dejar encargadas al sustituto -->
<!-- cada vez que se cambiar de activo a ausente, aparece y desaparece el textarea -->
<!-- según el estado del profesor la última vez que se editaron los datos, se mostrará el textarea o no-->
<div class="col-md-6">
    <label for="estado" class="form-label">Estado del profesor:</label>
    <select class="form-select" name="estado" id="estado" onchange="mostrarTA()">
        <?php if (strcmp($prof['estado'], "Activo") == 0) { ?>
            <option value="Activo" selected>Activo</option>
            <option value="Ausente">Ausente</option>
        </select>
</div><br>
    <div class="form-floating">
        <textarea class="form-control" name="tareas" style="display: none; height: 300px" id="tareas" value="<?php echo $prof['tareas_ausente']; ?>"><?= $prof['tareas_ausente']; ?></textarea>
        <label id="label" for="tareas" style="display: none">Tareas a realizar durante el periodo de ausencia: </label>
    </div><br>
    <!-- boton submint formulario -->
    <div class="d-flex justify-content-end bd-highlight mb-3">
        <button id="editar" type="submit" class="btn btn-primary" disabled>Editar</button>
    </div>
```

Figura 5.2.2.18 Código HTML formulario editar profesor (3)

En caso de que sea *ausente*, se mostrará el cuadro de texto relleno con las tareas que debe realizar el profesor que lo sustituya en su ausencia.

```
<?php } else { ?>
    <option value="Activo">Activo</option>
    <option value="Ausente" selected>Ausente</option>
    </select>
    </div><br>
    <div class="form-floating">
        <textarea class="form-control" name="tareas" style="display: block; height: 300px" id="tareas" value="<?php echo $prof['tareas_ausente']; ?>"><?= $prof['tareas_ausente']; ?></textarea>
        <label id="label" for="tareas" style="display: block">Tareas a realizar durante el periodo de ausencia: </label>
    </div>
    <div class="d-flex justify-content-end bd-highlight mb-3">
        <button id="editar" type="submit" class="btn btn-primary"disabled>Editar</button>
    </div>

<?php } ?>
```

Figura 5.2.2.19 Código HTML formulario editar profesor (3)

Si el usuario cambia el estado del profesor, el cuadro de texto aparecerá o desaparecerá en función del estado que se haya puesto, sin necesidad de actualizar la página ni enviarlo a ningún formulario. Esta acción se realiza con el siguiente código JavaScript.

```
function mostrarTA() {
    var select = document.getElementById('estado');
    select.addEventListener('change',
            function () {
                var select = this.options[select.selectedIndex];
            });
    if (select.value === "Ausente") { //si se ha seleccionado Ausente, se muestra el textarea
        document.getElementById('tareas').style.display = 'block';
        document.getElementById('label').style.display = 'block';
    } else { //si se ha seleccionado activo se oculta
        document.getElementById('tareas').style.display = 'none';
        document.getElementById('label').style.display = 'none';
    }
}
```

Figura 5.2.2.20 Función JavaScript que muestra el textarea

En el momento en el que el usuario modifique algún dato, el botón llamado *Editar* que se encuentra al final del formulario se habilitará. La función que habilita el botón, es un función JavaScript que detecta si hay algún cambio en los input del formulario y activa el botón.

```
<script>
    $(document).ready(function () {
        $('#form').on('input change', function () {
            $('#editar').attr('disabled', false);
        });
    })
</script>
```

Figura 5.2.2.21 Función JavaScript que habilita el botón

Por último, el usuario pulsará el botón y el sistema llamará al archivo *saveEditProf.php* que realizará la consulta UPDATE con los parámetros que se obtuvieron utilizando POST y actualizará los datos del profesor. Para saber qué profesor tenemos que actualizar, se pasará por URL el *dni* del profesor.

El sistema también obtendrá también todos los datos de los profesores que se han registrado en la aplicación para comprobar si hay alguno que coincida con el profesor cuyos datos se van a actualizar. Lo primero que comprueba es si ese correo pertenecía anteriormente al mismo usuario, es decir que no se ha editado, o si pertenece a otro usuario en caso de que se haya editado.

```php
$dni_p = $_GET['dni']; //coge el dni de la url
//selecciona todos los datos de la tabla profesor
$lprof = mysqli_query($connect, "SELECT * FROM profesor WHERE email ='" . $_POST['email'] . "'") or die(mysqli_connect_error());
$row = mysqli_fetch_assoc($lprof); //ejecuta la consulta
$found = 0;
$id = NULL;

//selecciona todos los datos de la tabla login
$lUsers = mysqli_query($connect, "SELECT * FROM login") or die(mysqli_connect_error());
$p = mysqli_fetch_assoc($lUsers);
$foundp = 0; //0 si no hay ningún profesor en la base de datos con ese correo, 1 si lo hay

if (mysqli_num_rows($lprof) > 0) {
    while ($found == 0 && $row != NULL) { //comprueba si ya hay alguien con ese correo y no es el mismo profesor
        if (strcasecmp($row['dni_profesor'], $dni_p) != 0) {
            $found = 1; //marca found a 1
        } else { //si no existe
            $row = mysqli_fetch_assoc($lprof); //coge el siguiente profesor de la base de datos
        }
    }
}
```

Figura 5.2.2.22 Comprobación de correo y DNI

En caso de que el correo no se haya editado o no pertenezca a otra persona en caso de edición, el sistema comprobará si ese usuario ya se ha registrado en la aplicación para obtener su *id_login* y así poder actualizar los roles de usuario en caso de que hayan sido editados.

```php
if ($found == 0) {
    $profe = $_POST['apellido1'] . " " . $_POST['apellido2'] . ", " . $_POST['nombre']; //combina el nombre del profesor en una variable
    //comprueba si hay un profesor en login con ese correo para relacionarlo con este profesor que se va a añadir
    while ($foundp == 0 && $p != NULL) {
        if (strcasecmp($p['email'], $_POST['email']) == 0) { //si en la base de datos ya esta el correo
            $foundp = 1; //pone found a 1
            $id = $p['id_login'];
        } else { //si no está el correo sigue buscando hasta que lo encuentra o no hay más donde buscar
            $p = mysqli_fetch_assoc($lUsers);
        }
    }
```

Figura 5.2.2.23 Comprobación de correo en login

Si el estado del profesor es *ausente* y no ha rellenado el cuadro de texto con las tareas, en el atributo tareas de la base de datos se guardará "No hay tareas". En caso de que el profesor se haya cambiado a activo, en el atributo tareas se guardará NULL.

```php
if (strcmp($_POST['estado'], "Ausente") == 0) { //si el estado del profesor es ausente
    if (strcmp($_POST['tareas'], "") == 0) { //si no ha dejado tareas para el profesor sustituto
        $tareas = "No hay tareas"; //se le asigna no hay tareas
    }
    $query = "UPDATE profesor SET profesor='" . $profe . "', apellido1='" . $_POST['apellido1'] . "', apellido2='" . $_POST['apellido2'] . "', nombre='" . $_POST['nombre'] . "', "
            . "telefono='" . $_POST['telefono'] . "', email='" . $_POST['email'] . "', estado='" . $_POST['estado'] . "', tareas_ausente='" . $tareas . "' WHERE dni_profesor='" . $dni_p . "'"; //c
} else { //si el profesor está activo
    $tareas = NULL; //a tareas se le asigna null
    $query = "UPDATE profesor SET profesor='" . $profe . "', apellido1='" . $_POST['apellido1'] . "', apellido2='" . $_POST['apellido2'] . "', nombre='" . $_POST['nombre'] . "', "
            . "telefono='" . $_POST['telefono'] . "', estado='" . $_POST['estado'] . "', tareas_ausente='" . $tareas . "', id_login='" . $id . "' WHERE dni_profesor='" . $dni_p . "'"; // consulta
}

echo $res = mysqli_query($connect, $query) or die(mysqli_connect_error() . 'error al editar el profesor'); //ejecuta la consulta
```

Figura 5.2.2.24 Consulta UPDATE profesor

En caso de que el usuario se haya registrado anteriormente en la aplicación, se procederá a actualizar los roles que el usuario tiene. Una vez actualizados los datos, el sistema volverá a la página principal de la gestión del profesorado y se mostrará un mensaje diciendo que se han editado los datos correctamente.

```php
    if ($foundp == 1) { //si el usuario ya se ha registrado, se actualizan sus roles y el correo
        $rol = $_POST['rol'][0] . " " . $_POST['rol'][1] . " " . $_POST['rol'][2];
        $q = "UPDATE login SET  rol='" . $rol . "', email='" . $_POST['email'] . "' WHERE id_login='" . $id . "'";
        echo $q;
        echo $rp = mysqli_query($connect, $q)or die(mysqli_connect_error() . 'error al update profesor id_login');
    }
    $urlr = $url . "gprofesor.php?ed=1" . "&id_login=" . $_GET['id_login']; //url a la que redirreciona
    header('Location: ' . $urlr); //redirecciona a gprofesor
} else {
    echo '<script> alert("Ya hay un profesor con este correo. Por favor utilice otro."); history.back();</script>'; //vuelve al formulario de añadir
}
```

Figura 5.2.2.25 Consulta UPDATE login

Como el resto de formularios de esta sección, también posee un botón llamado Volver a Gestión del profesorado el cual redirecciona a la página *gprofesor.php*.

A continuación se mostrará un ejemplo del formulario de edición de la unidad:

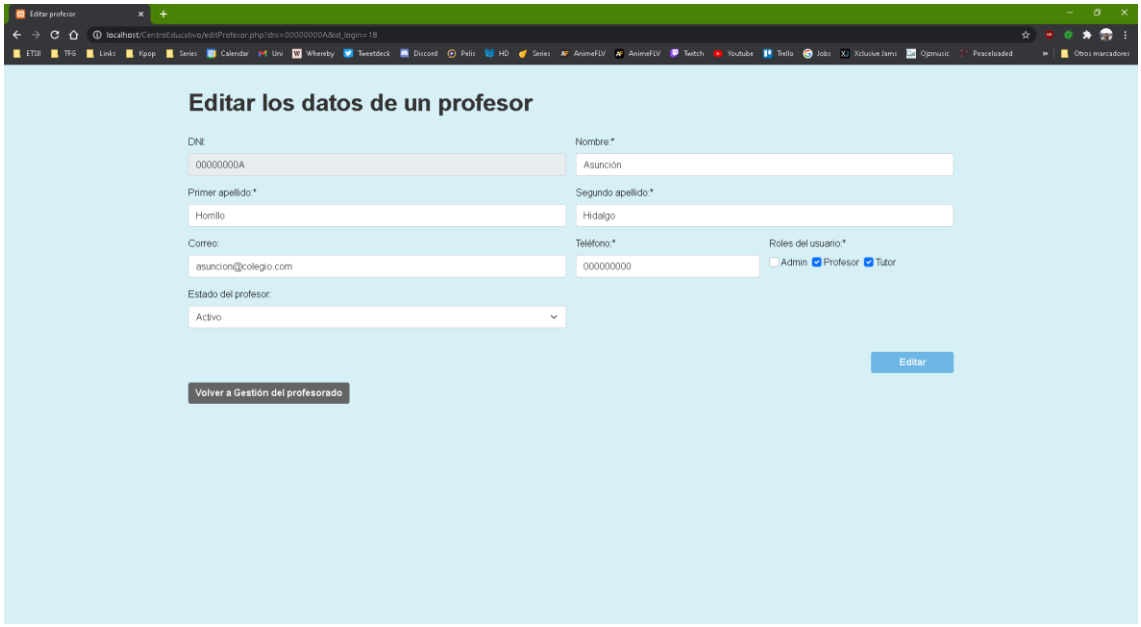

Figura 5.2.2.26 Formulario editar profesor

Es la segunda opción disponible para cada fila de datos de la tabla. Cuando el usuario pulsa el botón *Consultar*, el sistema muestra la página *showProfesor.php* la cual es un formulario que muestra los datos de ese profesoren concreto. Para ello se envía por URL el *dni* del profesor cuyos datos se desea consultar y el sistema realiza una consulta SELECT para obtener los datos de la base de datos. También se obtendrán los datos de inicio de sesión del usuario, para poder mostrar los roles en caso de que se hayan registrado anteriormente.

```php
$dni_p = $_GET['dni']; //coge el dni de la url
//obtiene todos los profesores de la base de datos
$query = mysqli_query($connect, 'SELECT * FROM profesor WHERE dni_profesor="' . $dni_p . '"') or die(mysqli_connect_error() . "no coge el dni");
$prof = mysqli_fetch_assoc($query);

$lUsers = mysqli_query($connect, "SELECT * FROM login WHERE id_login ='" . $prof['id_login'] . "'") or die(mysqli_connect_error());
$p = mysqli_fetch_assoc($lUsers);
```

Figura 5.2.2.27 Consulta SELECT profesor

Una vez se han obtenido los datos de la base de datos, el sistema muestra un formulario con los datos del profesor, en caso de que el estado del profesor sea ausente, se mostrará el cuadro de texto con las tareas que ha dejado. Estos datos no se pueden editar, solo se muestran a modo de consulta para el usuario.

```
<div class="container signup-form">
    <form class=" row g-3" method="POST" enctype="multipart/form-data">
        <div class="col-md-6">
            <label for="dni" class="form-label">DNI: </label>
            <input class="form-control" name="dni" type="text" readonly maxlength="9" value="<?php echo $prof['dni_profesor']; ?>"/>
        </div>
        <div class="col-md-6">
            <label for="nombre" class="form-label">Nombre: </label>
            <input class="form-control" name="nombre" type="text" readonly value="<?php echo $prof['nombre']; ?>"/>
        </div>
        <div class="col-md-6">
            <label for="apellido1" class="form-label">Primer apellido: </label>
            <input class="form-control" name="apellido1" type="text" readonly value="<?php echo $prof['apellido1']; ?>"/>
        </div>
        <div class="col-md-6">
            <label for="apellido2" class="form-label">Segundo apellido: </label>
            <input class="form-control" name="apellido2" type="text" readonly value="<?php echo $prof['apellido2']; ?>"/>
        </div>
        <div class="col-md-6">
            <label for="email" class="form-label">Correo: </label>
            <input class="form-control" name="email" type="email" readonly value="<?php echo $prof['email']; ?>"/>
        </div>
        <div class="col-md-3">
            <label for="telefono" class="form-label">Tel&eacute;fono</label>
            <input class="form-control" name="telefono" type="tel" readonly value="<?php echo $prof['telefono']; ?>"/>
        </div>
```

Figura 5.2.2.28 Código HTML formulario consultar profesor(1)

```
<?php if (mysqli_num_rows($lUsers) > 0) { ?>
    <div class="col-md-3">
        <label for="rol" class="form-label">Roles del usuario:* </label><br>
        <?php if (strpos($p['rol'], "1") !== false) { ?>
        <input type="checkbox" class="form-check-input" name="rol[]" readonly value="1" checked> Admin</input> 
        <?php } else { ?>
            <input type="checkbox" class="form-check-input" name="rol[]" readonly value="1"> Admin</input> 
        <?php } ?>
        <?php if (strpos($p['rol'], "2") !== false) { ?>
            <input type="checkbox" class="form-check-input" name="rol[]" readonly value="2" required checked> Profesor</input> 
        <?php } else { ?>
            <input type="checkbox" class="form-check-input" name="rol[]" readonly value="2" required> Profesor</input> 
        <?php } ?>
        <?php if (strpos($p['rol'], "3") !== false) { ?>
            <input type="checkbox" class="form-check-input" name="rol[]" readonly value="3" checked> Tutor</input> 
        <?php } else { ?>
            <input type="checkbox" class="form-check-input" name="rol[]" readonly value="3"> Tutor</input> 
        <?php } ?>
    </div>
<?php } ?>
<div class="col-md-6">
    <label for="estado" class="form-label">Estado del profesor:</label>
    <input class="form-control" type="text" name="estado" readonly value="<?php echo $prof['estado']; ?>"/>
</div>
<!-- en caso dde que el estado del profesor sea asuente, se mostrará el textarea con las tareas que ha dejado -->
<?php if (strcmp($prof['estado'], "Ausente") == 0) { ?>
    <div class="form-floating">
        <textarea class="form-control" style="display: block; height: 300px" name="tareas" readonly value="<?php echo $prof['tareas_ausente']; ?>"><?php echo $prof['tareas_ausente']; ?></textarea>
        <label class="form-label" for="tareas" style="display: block">Tareas para el periodo de ausencia: </label>
    </div>
<?php } ?>
```

Figura 5.2.2.29 Código HTML formulario consultar profesor(2)

Como el resto de formularios de esta sección, también posee un botón llamado Volver a Gestión del profesorado el cual redirecciona a la página *gprofesor.php*.

```
<div class="d-flex justify-content-start bd-highlight mb-3">
    <button role="link" class="btn btn-secondary" onclick="window.location = '<?php echo $url;?>gprofesor.php?id_login=<?php echo $_GET['id_login']; ?>'">Volver a Gesti&oacute;n del profesorado</button>
</div>
```

Figura 5.2.2.30 Código HTML botón volver atrás

A continuación se mostrará un ejemplo del formulario de consulta de la unidad:

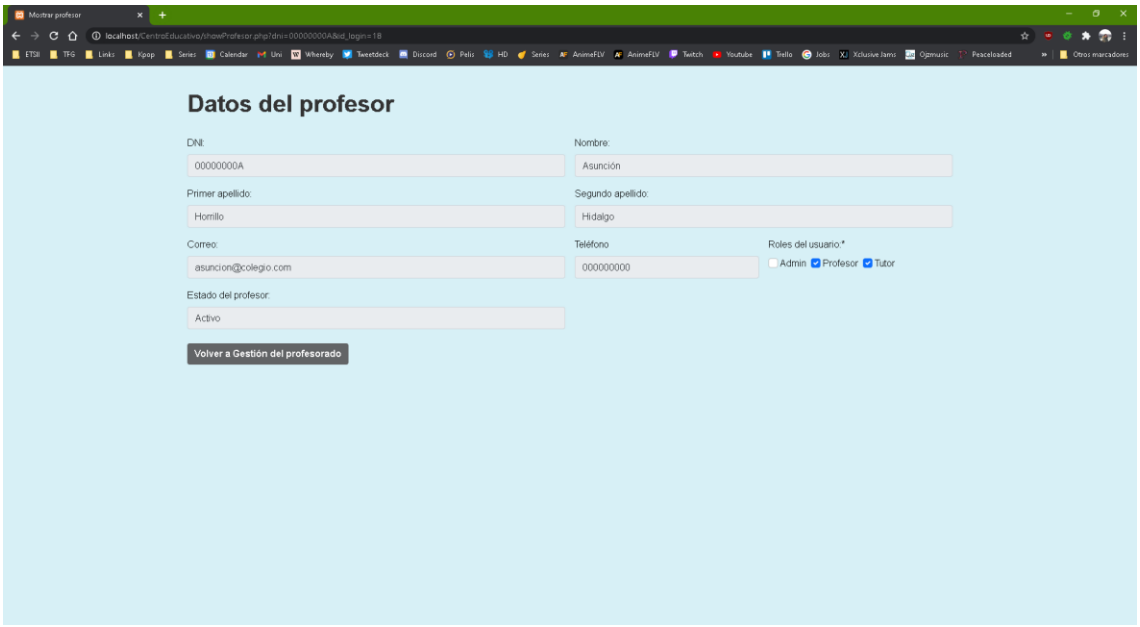

Figura 5.2.2.31 Formulario consultar profesor

La última acción a realizar sobre cada profesor es la de eliminar. El usuario pulsará el botón *Eliminar* y el sistema ejecutará el siguiente código JavaScript que lo que hace es mostrar una mensaje preguntando si el usuario realmente desea eliminar el registro. En caso de que el usuario cierre el mensaje o haga click en la opción *Cancel*, se cerrará el mensaje y el sistema no eliminará nada.

```
// funcion para borrar filas de la tabla
$(document).ready(function () {
    $('.delete').click(function () {
        var el = this;
        //coge la id del elemento a borrar
        var deleteid = $(this).data('id');
        // muestra caja de confirmación
        bootbox.confirm("¿Realmente quiere eliminar?", function (result) {
            if (result) {
                //borra el elemento y actualiza la tabla con ajax
                $.ajax({
                    url: 'lib/deleteProfesor.php',
                    type: 'POST',
                    data: {id: deleteid},
```

Figura 5.2.2.32 Código JavaScript eliminar profesor (1)

En caso de que el usuario haga click en la opción *Ok*, el sistema ejecutará el archivo *deleteProfesor.php*. En este archivo, el sistema recoge el *dni* del profesor que se quiere eliminar, se ejecuta la consulta DELETE y muestra el número 1 en caso de que se haya podido eliminar. Si no se pudo eliminar, el sistema mostrará el número 0.

```php
if (isset($_POST['id'])) { //si el id existe y no es null
    $dni_p = $_POST['id'];//obtiene el id del formulario

    $query = "DELETE FROM profesor WHERE dni_profesor='" . $dni_p . "'"; //consulta
    $res = mysqli_query($connect, $query) or die(mysqli_connect_error() . 'error al eliminar el profesor'); //ejecuta la consulta
    echo 1;
    exit;
} else {
    echo 0;
    exit;
}
```

Figura 5.2.2.33 Código PHP eliminar profesor

Si el sistema recibe el número 1 como respuesta, la fila que se ha eliminado de la base de datos se mostrará en rojo y desaparecerá de la tabla usando AJAX. En caso de que la respuesta haya sido 0, se mostrará un mensaje diciendo que no se pudo eliminar el registro.

```javascript
success: function (response) {
    //borra el elemento
    if (response == 1) {
        $(el).closest('tr').css('background', 'tomato');
        $(el).closest('tr').fadeOut(800, function () {
            $(this).remove();
        });
    } else { // si no se ha podido eleminar muestra el mensaje
        bootbox.alert('No se ha eliminado.');
    }
}
```

Figura 5.2.2.34Código JavaScript eliminar profesor (2)

Para terminar este apartado, justo debajo de la tabla hay un botón para volver a la página de inicio, en el caso de que el usuario quiera acceder a otro apartado de gestión.

```php
<div class="container">
    <button role="link" class="btn btn-secondary" onclick="window.location = '<?php echo $url; ?>home.php?id_login=<?php echo $_GET['id_login']; ?>'">Volver a la página de inicio</button>
</div>
```

Figura 5.2.2.35 Código HTML volver a inicio

5.2.3 Gestión de asignaturas

La página principal de la sección de Gestión de asignaturas es la siguiente:

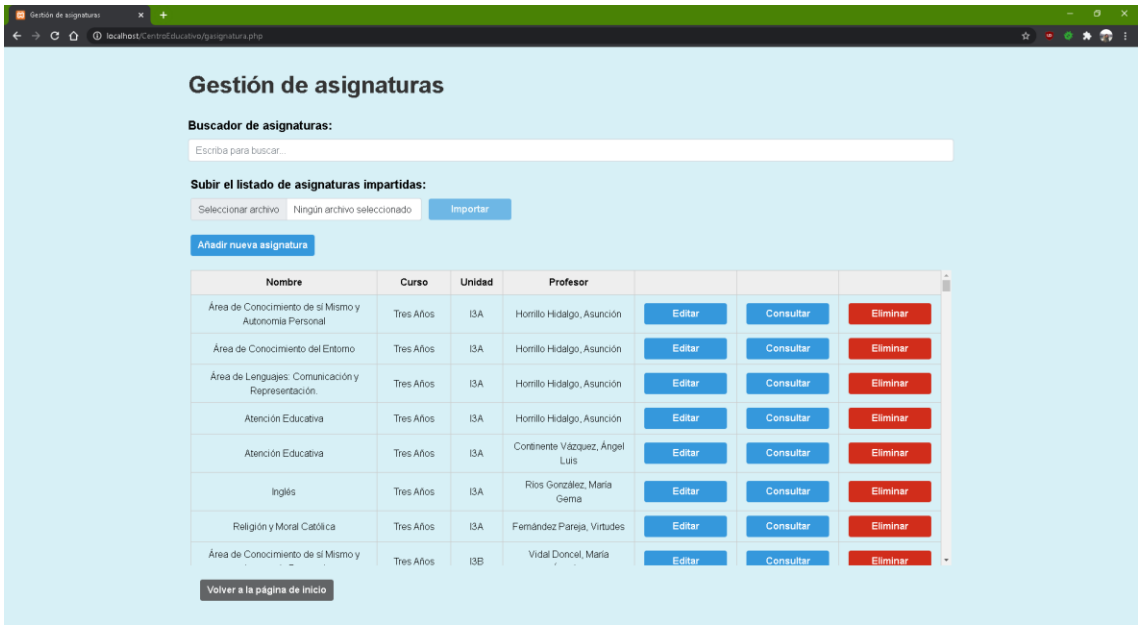

Figura 5.2.3.1 Página principal de asignaturas

En esta página, lo primero que encontramos es el buscador en el cual se puede buscar por nombre de asignatura, curso, unidad o tutor, es decir, si se quiere buscar todas las asignaturas que en su nombre contengan la palabra *Área* solo es necesario poner en el buscador la palabra *Área* y en la tabla aparecería la lista. Este buscador, va actualizando la lista de la tabla conforme el usuario va escribiendo, por lo que no es necesario pulsar ningún botón para buscar el texto introducido.

```
<!-- buscar una asignatura en la base de datos -->
<div class="container">
    <label for="buscador" class="form-label">Buscador de asignaturas: </label>
    <input class="form-control" id="buscador" type="text" autocomplete="off" placeholder="Escriba para buscar..." /><br>
</div>
```

Figura 5.2.3.2 Código HTML del buscador

La función que realiza la acción de buscar lo que hace es comprobar si alguna fila tiene algún dato que coincida con lo buscado, en cuyo caso la muestra, si no coincide la esconde.

```
$(document).ready(function () {
    $("#buscador").keyup(function () {
        _this = this;
        // muestra solo las filas que coinciden con lo buscado
        $.each($("#busqueda tbody tr"), function () {
            if ($(this).text().toLowerCase().indexOf($(_this).val().toLowerCase()) === -1)
                $(this).hide();
            else
                $(this).show();
        });
    });
});
```

Figura 5.2.3.3 Función JavaScript del buscador

Lo siguiente que encontramos en la página es el importador de archivos, en este caso de asignaturas, el usuario debe seleccionar una archivo pulsando el botón *Seleccionar archivo*, a continuación se abrirá el explorador de archivos del ordenador, el usuario escogerá el archivo que quiere importar y pulsará el botón *Importar*. Hasta que no se seleccione un archivo, el botón *Importar* estará deshabilitado, es decir, será visible pero no podrá pulsarse. La acción que realiza el sistema tras la pulsación del botón se ha detallado en el apartado 5.1.5.

Al igual que con el archivo que se importa en gestión de unidades, aquí tampoco hay ningún tipo de control sobre el número de veces que se puede subir el archivo, es decir, el sistema no controla que se suba la misma asignatura dos veces, ya que puede haber datos distintos como por ejemplo que haya dos asignaturas iguales pero que cambie el profesor que la imparte. En caso de que se haya subido dos veces la misma asignatura, el usuario puede buscarla y borrar la que quiera.

En este caso, también se controla quien puede importar o añadir datos y al igual que en el resto de apartados de gestión esta opción solo está disponible si el usuario tiene el rol de *admin*.

```
<?php if (strpos($_l['rol'], "1") !== false) { ?>
    <!-- Importar un conjunto de asignaturas a traves de un csv -->
    <div class="input-group row btn-toolbar">
        <form id="form" method="POST" enctype="multipart/form-data" action="File import/uploadMatUnidProf.php?id_login=<?php echo $_GET['id_login']; ?>">
            <label for="file" class="form-label">Subir el listado de asignaturas impartidas:</label><br>
            <div class="btn-group me-2 ">
                <input type="file" name="file" class="form-control"/>
            </div>
            <div class="btn-group me-2">
                <button id="importar" class="btn btn-primary" type="submit" disabled>Importar</button>
            </div>
        </form>
    </div><br>
```

Figura 5.2.3.4 Código HTML del importador

Si hubo errores al importar el archivo o todo se importó correctamente, el sistema mostrará un mensaje notificando al usuario lo que ha ocurrido. Esta acción se repetirá al añadir o editar los datos. Si la notificación es de color rojo, es que hubo errores, si la notificación es de color verde es que la acción se llevó a cabo con éxito.

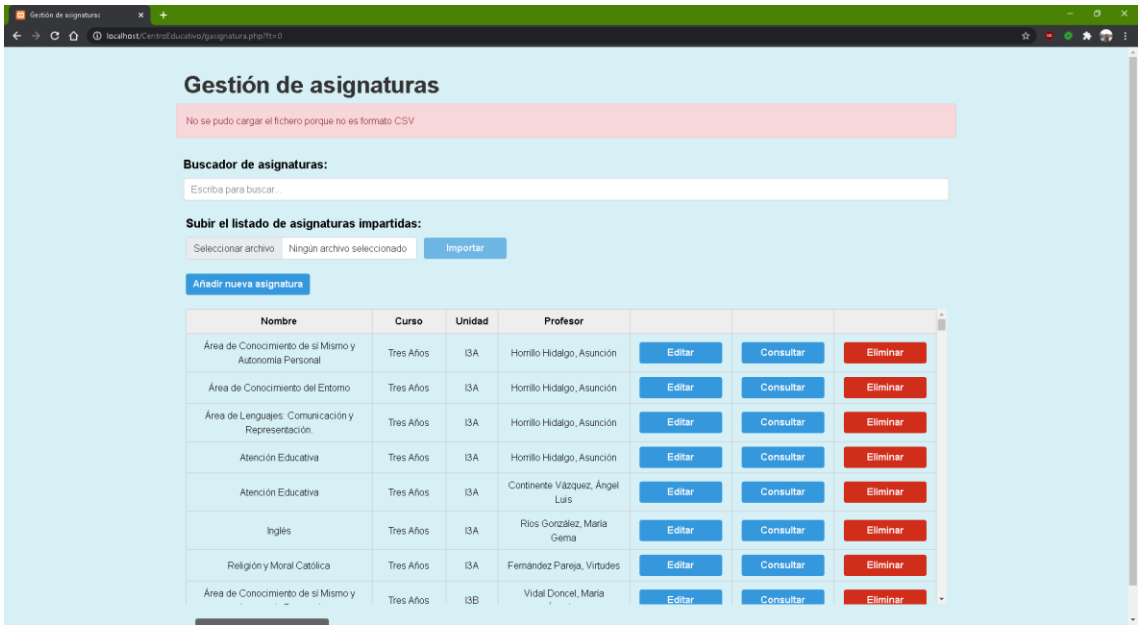

Figura 5.2.3.5 Página principal con alerta

Según el parámetro que se reciba por URL y el valor del parámetro, el sistema mostrará una alerta u otra. Si es *ft* el parámetro que recibe con valor 0 significa que hubo un error en el formato del archivo, si recibe el parámetro *ok* con valor 1 significa que el fichero se ha importado correctamente, si el parámetro es *add* con valor 1 significa que se han añadido los datos correctamente y si es *ed* con valor 1 es que se han editado los datos correctamente.

```php
<?php //mensajes de alerta para la subida de archivos y datos
if (isset($_GET['ft']) && $_GET['ft'] == 0) {
    echo '<div class="container alert alert-danger" role="alert">No se pudo cargar el fichero porque no es formato CSV</div>';
}
if (isset($_GET['ok']) && $_GET['ok'] == 1) {
    echo '<div class="container alert alert-success" role="alert">Se han importado los datos correctamente</div>';
}
if (isset($_GET['add']) && $_GET['add'] == 1) {
    echo '<div class="container alert alert-success" role="alert">Se ha añadido los datos correctamente</div>';
}
if (isset($_GET['ed']) && $_GET['ed'] == 1) {
    echo '<div class="container alert alert-success" role="alert">Se ha editado los datos correctamente</div>';
}
?>
```

Figura 5.2.3.6 Código PHP de alertas

La siguiente opción que encontramos en la página principal es la de añadir asignatura nueva, es un botón que lleva al usuario a un formulario llamado *addAsignatura.php*.. Al igual que la anterior, solo se mostrará si el usuario tiene el rol de admin.

```
<div class="col">
    <button name="nuevo" role="link" class="btn btn-primary" onclick="window.location = '<?php echo $url; ?>addAsignatura.php?id_login=<?php echo $_GET['id_login']; ?>'">Añadir nueva asignatura</button>
</div><br>
```

Figura 5.2.3.7 Código HTML botón añadir asignatura

En este formulario, el usuario tendrá que agregar los datos de la nueva asignatura que desea añadir y pulsar el botón *Añadir*. Si en la etiqueta del campo hay un asterisco (*) significa que el campo es obligatorio de rellenar, es decir, no se podrán guardar los datos a no ser que todos los campos con asterisco estén rellenos.

Como cada asignatura pertenece a una unidad y es impartido por un profesor, para evitar que el usuario tenga que introducir el nombre completo del profesor o la unidad a la que pertenece la asignatura estos campos serán una lista desplegable en la que se selecciona el dato. Para ello, es necesario obtener todos los profesores y todas las unidades que se han añadido en la base de datos, por lo tanto los profesores y las unidades deben de estar añadidas antes de insertar las asignaturas.

```php
<?php
$luni = mysqli_query($connect, "SELECT * FROM unidad") or die(mysqli_connect_error()); //obtiene las unidades
$u = mysqli_fetch_assoc($luni);
$lprof = mysqli_query($connect, "SELECT * FROM profesor") or die(mysqli_connect_error()); //obtiene todos los profesores
$p = mysqli_fetch_assoc($lprof);
?>
<!-- formulario para rellenar los datos de la asignatura -->
<div class="container signup-form">
    <form class=" row g-3" action="lib/saveAddAsignatura.php?id_login=<?=$_GET['id_login']?>" method="POST" enctype="multipart/form-data">
        <div class="col-md-5">
            <label for="nombre" class="form-label">Introduzca el nombre:* </label>
            <input name="nombre" class="form-control" type="text" required/>
        </div>
        <div class="col-md-3">
            <label for="cod_sist_calificacion" class="form-label">Introduzca el sistema de calificaci&oacute;n: </label>
            <input name="cod_sist_calificacion" class="form-control" type="text" />
        </div>
        <div class="col-md-4">
            <label for="curso" class="form-label">Introduzca el curso al que pertenece la asignatura:* </label>
            <input name="curso" class="form-control" type="text" required/>
        </div>
```

Figura 5.2.3.8 Código HTML formulario añadir asignatura (1)

```php
<div class="col-md-6">
    <label for="profesor" class="form-label">Seleccione el profesor:* </label>
    <select name="profesor" class="form-select" required>
        <option>Elige</option>
        <?php while ($p != NULL) { ?>
            <option value="<?php echo $p['profesor'] ?>"><?php echo $p['profesor'] ?></option>
            <?php
            $p = mysqli_fetch_assoc($lprof);
        }
        ?>
    </select>
</div>
<div class="col-md-6">
    <label for="unidad" class="form-label">Seleccione la unidad:* </label>
    <select name="unidad" class="form-select" required>
        <option>Elige</option>
        <?php while ($u != NULL) { ?>
            <option value="<?php echo $u['unidad'] ?>"><?php echo $u['unidad'] ?></option>
            <?php
            $u = mysqli_fetch_assoc($luni);
        }
        ?>
    </select>
</div>
<!-- boton submit formulario -->
<div class="d-flex justify-content-end bd-highlight mb-3">
    <button type="submit" class="btn btn-primary">A&ntilde;adir</button>
</div>
```

Figura 5.2.3.9 Código HTML formulario añadir asignatura (2)

El formulario también posee un botón llamado Volver a Gestión de asignaturas el cual redirecciona a la página *gasignatura.php*.

```php
<div class="d-flex justify-content-start bd-highlight mb-3">
    <button role="link" class="btn btn-secondary" onclick="window.location = '<?php echo $url;?>gasignatura.php?id_login=<?php echo $_GET['id_login']; ?>'">Volver a Gesti&oacute;n de asignaturas</button>
</div>
```

Figura 5.2.3.10 Código HTML botón volver

Tras pulsar el botón *Añadir* el sistema llama al archivo *saveAddAsignatura.php* que es el encargado de añadir a la base de datos. El sistema realiza la consulta con los datos obtenidos mediante el método POST y la ejecuta, por último redirige a *gasignatura.php* y manda por URL el parámetro *add* para que muestre la alerta correcta.

```
$query = sprintf('INSERT INTO materia (nombre, cod_sist_calificacion, curso, unidad, profesor) VALUES ("%s", "%s", "%s", "%s", "%s")',
        $_POST['nombre'], $_POST['cod_sist_calificacion'], $_POST['curso'], $_POST['unidad'], $_POST['profesor']); //consulta
echo $res = mysqli_query($connect, $query) or die(mysqli_connect_error() . 'error al insertar la asignatura'); //ejecuta la consulta

$urlr = $url."gasignatura.php?add=1"."&id_login=" . $_GET['id_login'];; //url a la que redireccionar
header('Location: ' . $urlr); //redirecciona a la url anterior
```

Figura 5.2.3.11 Consulta INSERT asignatura

A continuación se muestra el formulario tal y como lo vería el usuario:

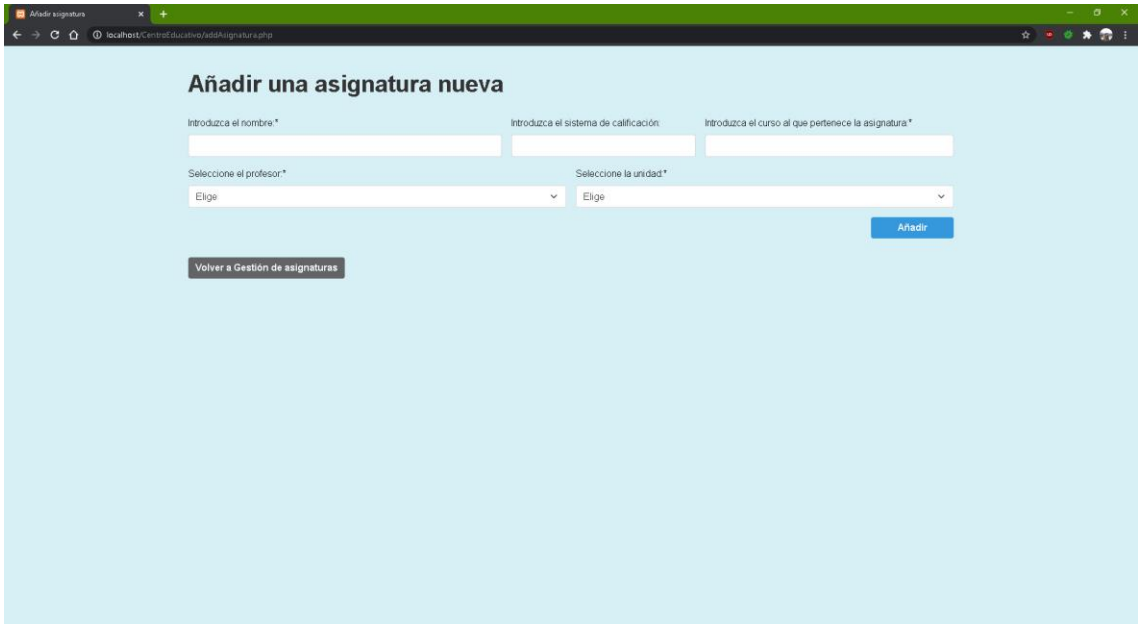

Figura 5.2.3.12 Formulario añadir asignatura

Tras el botón de Añadir nueva asignatura, aparece la tabla de datos. Para rellenar esta tabla, el sistema realiza una consulta SELECT que obtiene la lista de todas las asignaturas que se han introducido anteriormente. En caso de que la consulta no devuelva nada, la tabla solo mostrará la cabecera.

```
//obtiene todos las materias de la base de datos
$lmat = mysqli_query($connect, "SELECT * FROM materia") or die(mysqli_connect_error());
$row = mysqli_fetch_assoc($lmat);
```

Figura 5.2.3.13 Consulta SELECT asignaturas

Una vez tenemos la lista, por cada asignatura se mostrará una fila con el nombre de la asignatura, el curso y unidad al que pertenece y el profesor que la imparte, además de tres opciones que se detallarán más adelante.

```
<table>
    <thead>
        <tr>
            <th>Nombre</th>
            <th>Curso</th>
            <th>Unidad</th>
            <th>Profesor</th>
            <th></th>
            <th></th>
            <th></th>
        </tr>
    </thead>
    <?php while ($row != NULL) { ?>
        <tbody>
            <tr>
                <td><?php echo $row['nombre']; ?></td>
                <td><?php echo $row['curso']; ?></td>
                <td><?php echo $row['unidad']; ?></td>
                <td><?php echo $row['profesor']; ?></td>
```

Figura 5.2.3.14 Código HTML tabla de datos

Estas tres opciones, mencionadas anteriormente, que hacen uso de los datos son *Editar*, *Consultar* y *Eliminar*. Cada opción es un botón que, o bien te lleva a un formulario o bien realiza la acción que su propio nombre indica. Como en el caso de añadir, las acciones de editar y eliminar solo estarán disponibles para usuarios con el rol de administrador.

```
<?php if (strpos($l['rol'], "1") !== false) { ?>
    <td><a class="btn btn-primary" href='<?php echo $url; ?>editAsignatura.php?cod=<?php echo $row['codigo_materia']; ?>&id_login=<?php echo $_GET['id_login']; ?>'>Editar</a></td>
<?php } else { ?>
    <td><button class="btn btn-primary" disabled>Editar</button></td>
<?php } ?>
<td><a class="btn btn-primary" href='<?php echo $url; ?>showAsignatura.php?cod=<?php echo $row['codigo_materia']; ?>&id_login=<?php echo $_GET['id_login']; ?>'>Consultar</a></td>
<?php if (strpos($l['rol'], "1") !== false) { ?>
    <td><button class="delete btn btn-danger" id="<?= $row['codigo_materia'] ?>" data-id="<?= $row['codigo_materia'] ?>" >Eliminar</button></td>
<?php } else { ?>
    <td><button class="delete btn btn-danger" id="<?= $row['codigo_materia'] ?>" data-id="<?= $row['codigo_materia'] ?>" disabled>Eliminar</button></td>
<?php } ?>
```

Figura 5.2.3.15 Código HTML opciones de tabla

Si el usuario pulsa el botón Editar, se muestra la página *editAsignatura.php* a la cual se le ha enviado el *codigo_materia* de la asignatura que se quiere editar a través de la URL y el sistema realizará la consulta SELECT para obtener los datos. Además, se obtendrán todos los datos de las unidades y los profesores para las listas desplegables correspondientes.

```
$cod = $_GET['cod']; //coge el codigo de la asignatura de la url
//obtiene todos los datos de la asignatura de la base de datos
$query = mysqli_query($connect, 'SELECT * FROM materia WHERE codigo_materia="' . $cod . '"') or die(mysqli_connect_error() . "no coge el codigo");
$mat = mysqli_fetch_assoc($query);
$luni = mysqli_query($connect, "SELECT * FROM unidad") or die(mysqli_connect_error()); //obtiene las unidades
$u = mysqli_fetch_assoc($luni);
$lprof = mysqli_query($connect, "SELECT * FROM profesor") or die(mysqli_connect_error()); //obtiene todos los profesores
$p = mysqli_fetch_assoc($lprof);
```

Figura 5.2.3.16 Consulta SELECT de asignatura, unidad y profesor

Una vez que tenemos los datos de la asignatura, se realiza un formulario el cual ya está relleno.

```
<form id="form" class=" row g-3" action="lib/saveEditAsignatura.php?cod=<?= $mat['codigo_materia']; ?>&id_login=<?=$_GET['id_login']?>" method="POST" enctype="multipart/form-data">
    <div class="col-md-4">
        <label for="nombre" class="form-label">Nombre:* </label>
        <input class="form-control" name="nombre" type="text" required value="<?= $mat['nombre']; ?>"/>
    </div>
    <div class="col-md-4">
        <label for="cod_sist_calificacion" class="form-label">C&oacute;digo del sistema de calificaci&oacute;n: </label>
        <input class="form-control" name="cod_sist_calificacion" type="text" value="<?= $mat['cod_sist_calificacion']; ?>"/>
    </div>
    <div class="col-md-4">
        <label for="curso" class="form-label">Curso: </label>
        <input class="form-control" name="curso" type="text"  value="<?= $mat['curso']; ?>"/>
    </div>
```

Figura 5.2.3.17 Código HTML formulario editar asignatura (1)

En cuanto a los profesores y las unidades, la lista desplegable contendrá como opción marcada el profesor o la unidad que se introdujo cuando se añadió la asignatura a la base de datos. Si se quiere cambiar, el usuario solo debe desplegar la lista y elegir un profesor o unidad distinta.

```
<div class="col-md-6">
    <label for="profesor" class="form-label">Seleccione el profesor:* </label>
    <select name="profesor" class="form-select" required>
        <option>Elige</option>
        <?php
        while ($p != NULL) {
            if (strcmp($mat['profesor'], $p['profesor']) == 0) {
                ?>
                <option selected value="<?php echo $p['profesor'] ?>"><?php echo $p['profesor'] ?></option>
                <?php
            } else {
                ?>
                <option value="<?php echo $p['profesor'] ?>"><?php echo $p['profesor'] ?></option>
                <?php
            }
            $p = mysqli_fetch_assoc($lprof);
        }
        ?>
    </select>
</div>
<div class="col-md-6">
    <label for="unidad" class="form-label">Seleccione la unidad:* </label>
    <select name="unidad" class="form-select" required>
        <option>Elige</option>
        <?php while ($u != NULL) {
            if (strcmp($mat['unidad'], $u['unidad']) == 0) {
                ?>
                <option selected value="<?php echo $u['unidad'] ?>"><?php echo $u['unidad'] ?></option>
                <?php
            } else {
                ?>
                <option value="<?php echo $u['unidad'] ?>"><?php echo $u['unidad'] ?></option>
                <?php
            }
            $u = mysqli_fetch_assoc($luni);
        }
        ?>
    </select>
</div>
<div class="d-flex justify-content-end bd-highlight mb-3">
    <button id="editar" type="submit" class="btn btn-primary" disabled>Editar</button>
</div>
```

Figura 5.2.3.18 Código HTML formulario editar asignatura (2)

Al igual que en el formulario de edición de las unidades, en el momento en el que el usuario modifique algún dato, el botón llamado *Editar* que se encuentra al final del formulario se habilitará. La función que habilita el botón, es un función JavaScript que detecta si hay algún cambio en los input del formulario y activa el botón.

```
<script>
    $(document).ready(function () {
        $('#form').on('input change', function () {
            $('#editar').attr('disabled', false);
        });
    })
</script>
```

Figura 5.2.3.19 Función JavaScript que habilita el botón

Por último, el usuario pulsará el botón y el sistema llamará al archivo *saveEditAsignatura.php* que realizará la consulta UPDATE con los parámetros que se obtuvieron utilizando POST y actualizará los datos de la unidad. Para saber qué asignatura tenemos que actualizar, se pasará por URL el *codigo_materia* de la asignatura.

Una vez actualizados los datos, el sistema volverá a la página principal de la gestión de asignaturas y se mostrará un mensaje diciendo que se han editado los datos correctamente.

```
$cod = $_GET['cod']; //coge el dni de la url
$query = "UPDATE materia SET nombre='" . $_POST['nombre'] . "', cod_sist_calificacion='" . $_POST['cod_sist_calificacion'] . "', curso='" . $_POST['curso']
      ."', unidad='" . $_POST['unidad']."', profesor='" . $_POST['profesor']. "' WHERE codigo_materia='" . $cod . "'"; // consulta
echo $res = mysqli_query($connect, $query) or die(mysqli_connect_error() . 'error al editar la asignatura'); //ejecuta la consulta

$urlr = $url."gasignatura.php?ed=1"."&id_login=" . $_GET['id_login']; //url a la que redirreciona
header('Location: ' . $urlr); //redirecciona a gasignatura
```

Figura 5.2.3.20 Consulta UPDATE asignatura

Como el resto de formularios de esta sección, también posee un botón llamado Volver a Gestión de asignaturas el cual redirecciona a la página *gasignatura.php*.

A continuación se mostrará un ejemplo del formulario de edición de la asignatura:

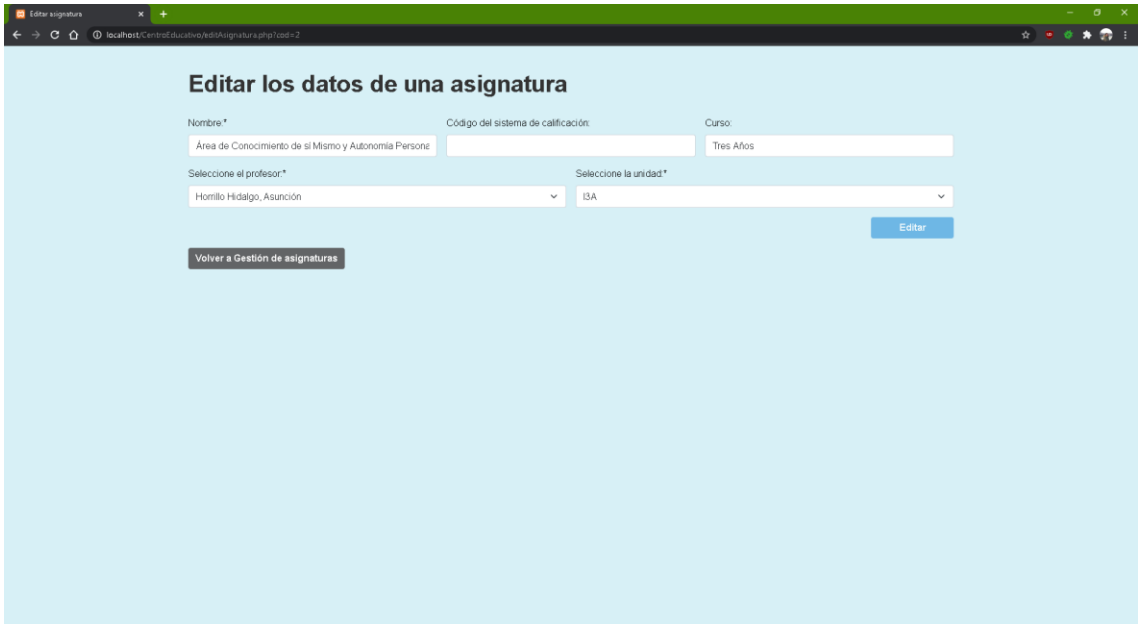

Figura 5.2.3.21 Formulario editar asignatura

Es la segunda opción disponible para cada fila de datos de la tabla. Cuando el usuario pulsa el botón *Consultar*, el sistema muestra la página *showAsignatura.php* la cual es un formulario que muestra los datos de esa asignatura en concreto. Para ello se envía por URL el *codigo_materia* de la asignatura que se desea consultar y el sistema realiza una consulta SELECT para obtener los datos de la base de datos.

```php
$cod = $_GET['cod']; //coge el codigo de la asignatura de la url
//obtiene todos los datos de la asignatura de la base de datos
$query = mysqli_query($connect, 'SELECT * FROM materia WHERE codigo_materia="' . $cod . '"') or die(mysqli_connect_error() . "no coge el codigo");
$mat = mysqli_fetch_assoc($query);
```

Figura 5.2.3.22 Consulta SELECT asignatura

Una vez se han obtenido los datos de la base de datos, el sistema muestra un formulario con los datos de la asignatura. Estos datos no se pueden editar, solo se muestran a modo de consulta para el usuario.

```php
<form id="form" class=" row g-3" enctype="multipart/form-data">
    <div class="col-md-4">
        <label for="nombre" class="form-label">Nombre: </label>
        <input class="form-control" name="nombre" type="text" required readonly disabled value="<?= $mat['nombre']; ?>"/>
    </div>
    <div class="col-md-4">
        <label for="cod_sist_calificacion" class="form-label">c&oacute;digo del sistema de calificaci&oacute;n: </label>
        <input class="form-control" name="cod_sist_calificacion" type="text" readonly disabled required value="<?= $mat['cod_sist_calificacion']; ?>"/>
    </div>
    <div class="col-md-4">
        <label for="curso" class="form-label">Curso: </label>
        <input class="form-control" name="curso" type="text"  readonly disabled value="<?= $mat['curso']; ?>"/>
    </div>
    <div class="col-md-6">
        <label for="profesor" class="form-label">Profesor: </label>
        <input class="form-control" name="profesor" type="text"  readonly disabled value="<?= $mat['profesor']; ?>"/>
    </div>
    <div class="col-md-6">
        <label for="unidad" class="form-label">Unidad: </label>
        <input class="form-control" name="unidad" type="text"  readonly disabled value="<?= $mat['unidad']; ?>"/>
    </div>
</form><br>
```

Figura 5.2.3.23 Código HTML formulario consultar asignatura

Como el resto de formularios de esta sección, también posee un botón llamado Volver a Gestión de asignaturas el cual redirecciona a la página *gasignatura.php*.

```php
<div class="d-flex justify-content-start bd-highlight mb-3">
    <button role="link" class="btn btn-secondary" onclick="window.location = '<?php echo $url;?>gasignatura.php?id_login=<?php echo $_GET['id_login']; ?>'">Volver a Gesti&oacute;n de asignaturas</button>
</div>
```

Figura 5.2.3.24 Código HTML botón volver atrás

A continuación se mostrará un ejemplo del formulario de consulta de la asignatura:

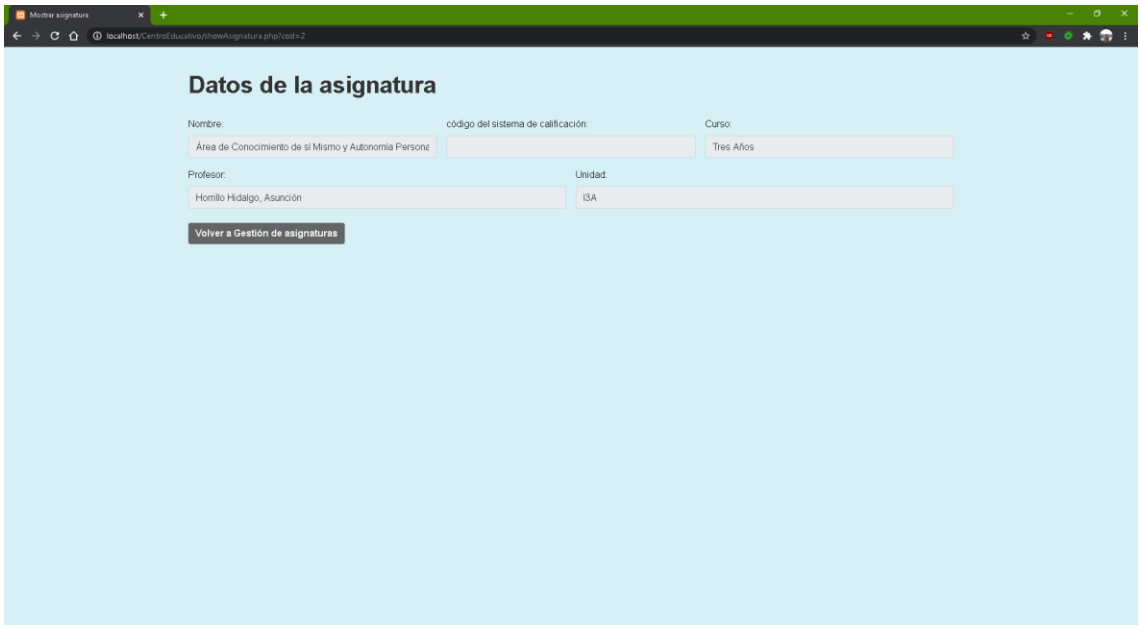

Figura 5.2.3.25 Formulario consultar asignatura

La última acción a realizar sobre cada asignatura es la de eliminar. El usuario pulsará el botón *Eliminar* y el sistema ejecutará el siguiente código JavaScript que lo que hace es mostrar una mensaje preguntando si el usuario realmente desea eliminar el registro. En caso de que el usuario cierre el mensaje o haga click en la opción *Cancel*, se cerrará el mensaje y el sistema no eliminará nada.

```
// funcion para borrar filas de la tabla
$(document).ready(function () {
    $('.delete').click(function () {
        var el = this;
        //coge la id del elemento a borrar
        var deleteid = $(this).data('id');
        // muestra caja de confirmación
        bootbox.confirm("¿Realmente quiere eliminar?", function (result) {
            if (result) {
                //borra el elemento y actualiza la tabla con ajax
                $.ajax({
                    url: 'lib/deleteAsignatura.php',
                    type: 'POST',
                    data: {id: deleteid},
```

Figura 5.2.3.26 Código JavaScript eliminar asignatura (1)

En caso de que el usuario haga click en la opción *Ok*, el sistema ejecutará el archivo *deleteAsignatura.php*. En este archivo, el sistema recoge la id de la unidad que se quiere eliminar, se ejecuta la consulta DELETE y muestra el número 1 en caso de que se haya podido eliminar. Si no se pudo eliminar, el sistema mostrará el número 0.

```php
if (isset($_POST['id'])) { //si el id existe y no es null
    $cod = $_POST['id'];//obtiene el id del formulario
    $query = "DELETE FROM materia WHERE codigo_materia='" . $cod . "'"; //consulta
    //ejecuta la consulta
    $res = mysqli_query($connect, $query) or die(mysqli_connect_error() . 'error al eliminar la asignatura');
    echo 1;
    exit;
} else {
    echo 0;
    exit;
}
```

Figura 5.2.3.27 Código PHP eliminar asignatura

Si el sistema recibe el número 1 como respuesta, la fila que se ha eliminado de la base de datos se mostrará en rojo y desaparecerá de la tabla usando AJAX. En caso de que la respuesta haya sido 0, se mostrará un mensaje diciendo que no se pudo eliminar el registro.

```javascript
success: function (response) {
    //borra el elemento
    if (response == 1) {
        $(el).closest('tr').css('background', 'tomato');
        $(el).closest('tr').fadeOut(800, function () {
            $(this).remove();
        });
    } else { // si no se ha podido eleminar muestra el mensaje
        bootbox.alert('No se ha eliminado.');
    }
}
```

Figura 5.2.3.28 Código JavaScript eliminar asignatura (2)

Para terminar este apartado, justo debajo de la tabla hay un botón para volver a la página de inicio, en el caso de que el usuario quiera acceder a otro apartado de gestión.

```html
<!-- boton para volver a home.php -->
<div class="container">
    <button role="link" class="btn btn-secondary" onclick="window.location = 'http://localhost/CentroEducativo/home.php'">Volver a la página de inicio</button>
</div>
```

Figura 5.2.3.29 Código HTML volver a inicio

5.3 Tercera iteración

Esta última iteración corresponde al desarrollo del CRUD del alumnado, se ha introducido el módulo de los informes de tutoría, el avance automático de curso del alumnado y el diseño completo y la incorporación de CSS y Bootstrap a la aplicación web. Al igual que en la segunda iteración, estas iteraciones no se habrían podido crear antes que las anteriores ya que el alumno pertenece a una unidad y los informes de los alumnos los realiza un profesor sobre una asignatura. Además, aquí también se tiene en cuenta el rol del usuario que ha iniciado sesión para poder acotar las actividades que puede realizar cada persona registrada.

5.3.1 Gestión del alumnado

La página principal de la sección de Gestión del alumnado es la siguiente:

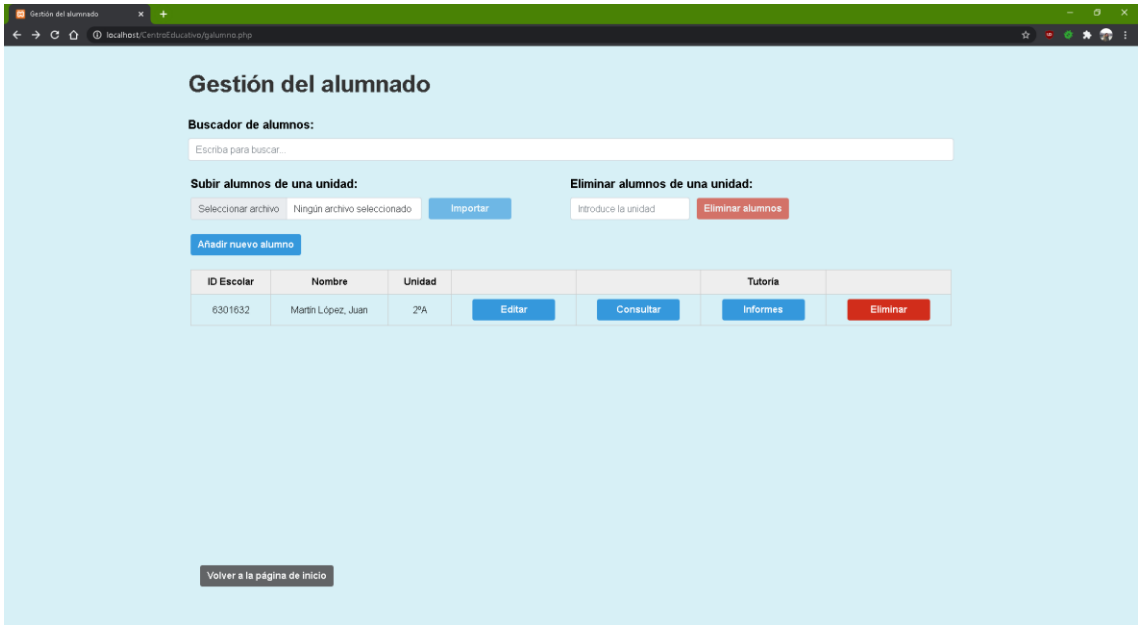

Figura 5.3.1.1 Página principal del alumnado

En esta página, lo primero que encontramos es el buscador en el cual se puede buscar por nombre, ID escolar o unidad, es decir, si se quiere buscar todos los alumnos de infantil de tres años solo es necesario poner en el buscador la palabra I3A y en la tabla aparecería la lista. Este buscador, va actualizando la lista de la tabla conforme el usuario va escribiendo, por lo que no es necesario pulsar ningún botón para buscar el texto introducido.

```
<div class="container"> <!-- buscar un alumno en la base de datos -->
    <label for="buscador" class="form-label">Buscador de alumnos: </label>
    <input class="form-control" id="buscador" type="text" autocomplete="off" placeholder="Escriba para buscar..." /><br>
</div>
```

Figura 5.3.1.2 Código HTML del buscador

La función que realiza la acción de buscar lo que hace es comprobar si alguna fila tiene algún dato que coincida con lo buscado, en cuyo caso la muestra, si no coincide la esconde.

```
$(document).ready(function () {
    $("#buscador").keyup(function () {
        _this = this;
        // muestra solo las filas que coinciden con lo buscado
        $.each($("#busqueda tbody tr"), function () {
            if ($(this).text().toLowerCase().indexOf($(_this).val().toLowerCase()) === -1)
                $(this).hide();
            else
                $(this).show();
        });
    });
});
```

Figura 5.3.1.3 Función JavaScript del buscador

Lo siguiente que encontramos en la página es el importador de archivos, en este caso de alumnos de una unidad, el usuario debe seleccionar una archivo pulsando el botón *Seleccionar archivo*, a continuación se abrirá el explorador de archivos del ordenador, el usuario escogerá el archivo que quiere importar y pulsará el botón *Importar*. Hasta que no se seleccione un archivo, el botón *Importar* estará deshabilitado, es decir, será visible pero no podrá pulsarse. La acción que realiza el sistema tras la pulsación del botón se ha detallado en el apartado 5.1.4. Esta acción solo la podrá realizar un usuario con rol de administrador.

```php
<?php if (strpos($_l['rol'], "1") !== false) { ?>
    <!-- Importar un conjunto de alumnos de una unidad a traves de un csv -->
    <div class="input-group row btn-toolbar">
        <div class="col-md-6">
            <form id="form" method="POST" enctype="multipart/form-data" action="File import/uploadAlumno.php?id_login=<?php echo $_GET['id_login']; ?>">
                <label for="file" class="form-label">Subir alumnos de una unidad:</label><br>
                <div class="btn-group me-2 ">
                    <input type="file" name="file" class="form-control"/>
                </div>
                <div class="btn-group me-2">
                    <button id="importar" class="btn btn-primary" type="submit" disabled>Importar</button>
                </div>
            </form>
        </div>
```

Figura 5.3.1.4 Código HTML del importador

Si hubo errores al importar el archivo o todo se importó correctamente, el sistema mostrará un mensaje notificando al usuario lo que ha ocurrido. Esta acción se repetirá al añadir o editar los datos. Si la notificación es de color rojo, es que hubo errores, si la notificación es de color verde es que la acción se llevó a cabo con éxito.

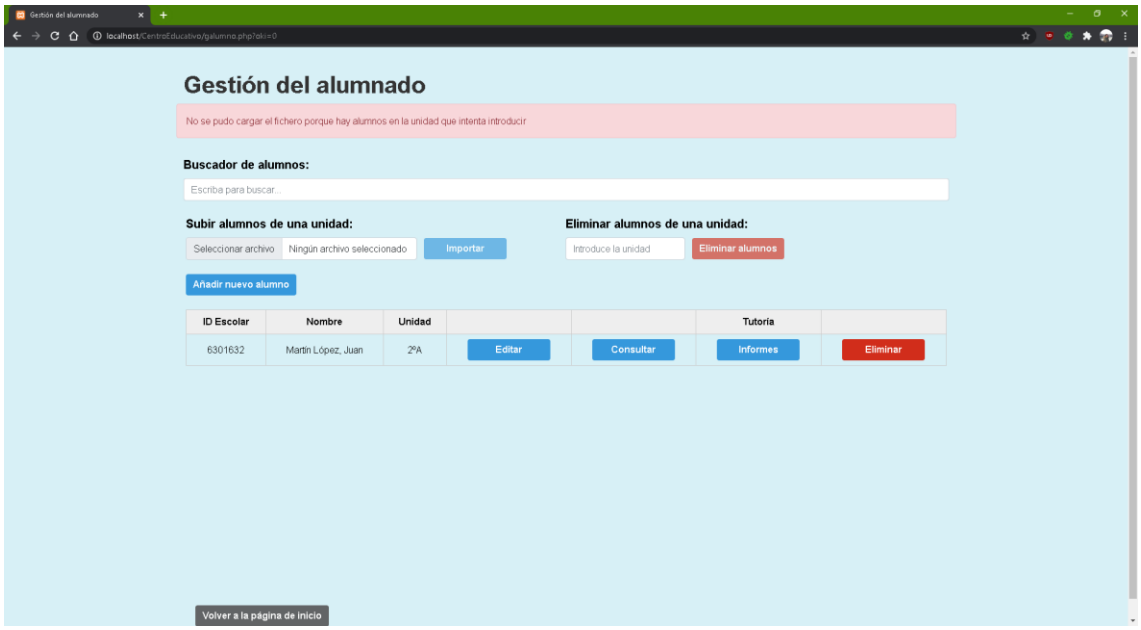

Figura 5.3.1.5 Página principal con alerta

Según el parámetro que se reciba por URL y el valor del parámetro, el sistema mostrará una alerta u otra. Si recibe *ok* con el valor 0 significa que no se pudo importar el fichero porque el alumno ya está guardado en la base de datos, si recibe el parámetro *oki* con valor 0 significa que ya hay alumnos en la base de datos que pertenecen a la misma unidad la cual se intenta introducir por lo que no pudo importarse el fichero.

Si recibe *ft* el parámetro que recibe con valor 0 significa que hubo un error en el formato del archivo, si recibe el parámetro *ok* con valor 1 significa que el fichero se ha importado correctamente, si el parámetro es *add* con valor 1 significa que se han añadido los datos correctamente y si es *ed* con valor 1 es que se han editado los datos correctamente. Por último, si recibe el parámetro *el* con valor 1 significa que se eliminaron los alumnos de la unidad correctamente y si lo recibe con valor 0 es porque no hay alumnos que pertenezcan a la unidad cuyos alumnos se quiere borrar.

```php
<?php //mensajes de alerta para la subida de ficheros y datos
if (isset($_GET['ok']) && $_GET['ok'] == 0) {
    echo '<div class="container alert alert-danger" role="alert">No se pudo cargar el fichero porque el alumno ya está guardado</div>';
}
if (isset($_GET['oki']) && $_GET['oki'] == 0) {
    echo '<div class="container alert alert-danger" role="alert">No se pudo cargar el fichero porque hay alumnos en la unidad que intenta introducir</div>';
}
if (isset($_GET['ft']) && $_GET['ft'] == 0) {
    echo '<div class="container alert alert-danger" role="alert">No se pudo cargar el fichero porque no es formato CSV</div>';
}
if(isset($_GET['ok']) && $_GET['ok'] == 1){
   echo '<div class="container alert alert-success" role="alert">Se han importado los datos correctamente</div>';
}
if(isset($_GET['add']) && $_GET['add'] == 1){
   echo '<div class="container alert alert-success" role="alert">Se ha añadido los datos correctamente</div>';
}
if(isset($_GET['ed']) && $_GET['ed'] == 1){
   echo '<div class="container alert alert-success" role="alert">Se ha editado los datos correctamente</div>';
}
if(isset($_GET['el']) && $_GET['el'] == 0){
   echo '<div class="container alert alert-warning" role="alert">No hay alumnos de esa unidad que eliminar</div>';
}
    if(isset($_GET['el']) && $_GET['el'] == 1){
   echo '<div class="container alert alert-success" role="alert">Se han eliminado los alumnos de la unidad correctamente</div>';
}
?>
```

Figura 5.3.1.6 Código PHP de alertas

En la misma fila que la importación alumnos de una unidad, se encuentra la eliminación de alumnos de una unidad. Al igual que la opción de importar, esta tampoco estará disponible si el usuario no es administrador. Además, antes de eliminar cualquier dato, le preguntará al usuario si desea eliminar todos los alumnos de la unidad.

```html
<div class="col-md-6">
    <form id="forme" method="POST" enctype="multipart/form-data" action="lib/deleteUnidadAlumno.php" onsubmit="return confirm('¿Realmente desea eliminar todos los alumnos de esta unidad?');">
        <label for="eliminar" class="form-label">Eliminar alumnos de una unidad:</label><br>
        <div class="btn-group me-2 ">
            <input type="text" name="eliminar" class="form-control" placeholder="Introduce la unidad"/>
        </div>
        <div class="btn-group me-2">
            <button id="eliminar" class="btn btn-danger" disabled>Eliminar alumnos</button>
        </div>
    </form>
</div>
```

Figura 5.3.1.7 Código HTML botón eliminar alumnos unidad

En esta opción solo será necesario introducir una unidad, por ejemplo 2ºA, y el sistema habilitará el botón *Eliminar alumnos* mediante este código JavaScript.

```javascript
//funcion para que se active el boton eliminar alumnos
//si hay algún cambio
$(document).ready(function () {
    $('#forme').on('input change', function () {
        $('#eliminar').attr('disabled', false);
    });
})
```

Figura 5.3.1.8 Código JavaScript habilitar botón eliminar alumnos

Cuando el usuario pulse el botón, el sistema mostrará un mensaje para confirmar si el usuario quiere eliminar realmente los alumnos de la unidad. En caso de que pulse *Aceptar*, el sistema ejecutará el archivo *deleteUnidadAlumno.php*, cuya función es realizar la consulta que elimina todos los alumnos cuyo atributo unidad coincide con la unidad que introdujo el usuario anteriormente. Tras la consulta, redirige a la página principal de gestión del alumnado y envía el parámetro el para que muestre la alerta de que se ha eliminado correctamente. En caso de que no haya alumnos que pertenezcan a esa unidad, el sistema vuelve a la página principal de gestión del alumnado y muestra una notificación amarilla avisando al usuario que no hay alumnos en esa unidad.

```php
$unidad = $_POST['eliminar']; //obtiene la unidad del formulario
$al = mysqli_query($connect, "SELECT * FROM alumno WHERE unidad='" . $unidad . "'");

if (mysqli_num_rows($al) > 0) { //si hay alumnos en la unidad
    $query = "DELETE FROM alumno WHERE unidad='" . $unidad . "'"; //consulta
    echo $res = mysqli_query($connect, $query) or die(mysqli_connect_error() . 'error al eliminar el alumno');
    $urlr = $url."galumno.php?el=1"."&id_login=" . $_GET['id_login']; //url a la que redireccionar
    header('Location: ' . $urlr); //redirecciona a la url anterior
} else { //si no hay alumnos en la unidad
    $urlr = $url."galumno.php?el=0"."&id_login=" . $_GET['id_login']; //url a la que redireccionar
    header('Location: ' . $urlr); //redirecciona a la url anterior F
}
```

Figura 5.3.1.9 Consulta DELETE alumnos de unidad

La siguiente opción que encontramos en la página principal es la de añadir nuevo alumno, es un botón que lleva al usuario a un formulario llamado *addAlumno.php*. Esta opción tampoco estará visible si tu rol es de profesor o de tutor.

```php
<!-- añadir un alumno a la base de datos -->
<div class="btn-toolbar">
    <div class="btn-group me-2">
        <button name="nuevo" role="link" class="btn btn-primary" onclick="window.location = '<?php echo $url; ?>addAlumno.php&id_login=<?php echo $_GET['id_login']; ?>'">Añadir nuevo alumno</button>
    </div>
</div><br>
```

Figura 5.3.1.10 Código HTML botón añadir alumno

En este formulario, el usuario tendrá que agregar los datos del alumno nuevo que desea añadir y pulsar el botón *Añadir*. Si en la etiqueta del campo hay un asterisco (*) significa que el campo es obligatorio de rellenar, es decir, no se podrán guardar los datos a no ser que todos los campos con asterisco estén rellenos.

Además, se obtendrá de la base de datos un listado de las unidades para que el usuario pueda seleccionar a la que pertenece el alumno. El ID escolar tiene como máximo una longitud de 10, es decir, que puede tener hasta 10 cifras. El campo *Sexo* tanto del alumno como de los tutores es una lista desplegable con los argumentos *H* o *M*.

```
<form class=" row g-3" action="lib/saveAddAlumno.php?id_login=<?=$_GET['id_login']?>" method="POST" enctype="multipart/form-data">
    <div class="col-md-4">
        <label for="idescolar" class="form-label">Introduzca el ID escolar:* </label>
        <input name="idescolar" class="form-control" type="text" required maxlength="10"/>
    </div>
    <div class="col-md-4">
        <label for="estadomat" class="form-label">Introduzca el estado de la matricula: </label>
        <input name="estadomat" class="form-control" type="text"/>
    </div>
    <div class="col-md-4">
        <label for="dni" class="form-label">Introduzca el DNI: </label>
        <input name="dni" class="form-control" type="text" maxlength="9"/>
    </div>
    <div class="col-md-6">
        <label for="nombre" class="form-label">Introduzca el nombre:* </label>
        <input name="nombre" class="form-control" type="text" required/>
    </div>
    <div class="col-md-3">
        <label for="fechanac" class="form-label">Introduzca el fecha de nacimiento:* </label>
        <input name="fechanac" class="form-control" type="date" required/>
    </div>
    <div class="col-md-3">
        <label for="sexo" class="form-label">Introduzca el sexo:* </label>
        <select name="sexo" class="form-select" required>
            <option value="H">H</option>
            <option value="M">M</option>
        </select>
    </div>
    <div class="col-md-6">
        <label for="apellido1" class="form-label">Introduzca el primer apellido:* </label>
        <input name="apellido1" class="form-control" type="text" required/>
    </div>
    <div class="col-md-6">
        <label for="apellido2" class="form-label">Introduzca el segundo apellido:* </label>
        <input name="apellido2" class="form-control" type="text" required/>
    </div>
```

Figura 5.3.1.11 Código HTML formulario añadir alumno (1)

Los campos como el correo o los teléfonos controlarán que se introduzcan correctamente los datos, es decir, que el teléfono esté formado por 9 cifras y que el correo esté escrito completamente.

```
<div class="col-md-12">
    <label for="direccion" class="form-label">Introduzca el direcci&oacute;n:* </label>
    <input name="direccion" class="form-control" type="text" required/>
</div>
<div class="col-md-6">
    <label for="localidad" class="form-label">Introduzca el localidad:* </label>
    <input name="localidad" class="form-control" type="text" required/>
</div>
<div class="col-md-4">
    <label for="provincia" class="form-label">Introduzca el provincia:* </label>
    <input name="provincia" class="form-control" type="text" required/>
</div>
<div class="col-md-2">
    <label for="codpostal" class="form-label">Introduzca el c&oacute;digo postal:* </label>
    <input name="codpostal" class="form-control" type="text" required maxlength="5"/>
</div>
<div class="col-md-4">
    <label for="localidadnac" class="form-label">Introduzca la localidad de nacimiento:* </label>
    <input name="localidadnac" class="form-control" type="text" required/>
</div>
<div class="col-md-4">
    <label for="provincianac" class="form-label">Introduzca el provincia de nacimiento:* </label>
    <input name="provincianac" class="form-control" type="text" required/>
</div>
<div class="col-md-4">
    <label for="paisnac" class="form-label">Introduzca la pa&iacute;s de nacimiento:* </label>
    <input name="paisnac" class="form-control" type="text" required/>
</div>
<div class="col-md-4">
    <label for="nacionalidad" class="form-label">Introduzca la nacionalidad:* </label>
    <input name="nacionalidad" class="form-control" type="text" required/>
</div>
<div class="col-md-4">
    <label for="telefono" class="form-label">Introduzca el tel&eacute;fono:* </label>
    <input name="telefono" class="form-control" type="tel" placeholder="666666666" required />
</div>
<div class="col-md-4">
    <label for="tlfsos" class="form-label">Introduzca el tel&eacute;fono de emergencia:* </label>
    <input name="tlfsos" class="form-control" type="tel" placeholder="666666666" required />
</div>
<div class="col-md-12">
    <label for="email" class="form-label">Introduzca el correo:* </label>
    <input name="email" class="form-control" type="email" placeholder="correo@servidor.com" required/>
</div>
```

Figura 5.3.1.12 Código HTML formulario añadir alumno (2)

Para seleccionar la unidad a la que pertenece el alumno, el usuario dispondrá de una lista desplegable en la que aparecerán todas las unidades insertadas en la base de datos.

```
<div class="col-md-4">
    <label for="numexp" class="form-label">Introduzca el n&uacute;mero expediente:* </label>
    <input name="numexp" class="form-control" type="text" required/>
</div>
<div class="col-md-4">
    <label for="curso" class="form-label">Introduzca el curso:* </label>
    <input name="curso" class="form-control" type="text" required/>
</div>
<div class="col-md-4">
    <label for="unidad" class="form-label">Introduzca el unidad:* </label>
    <select name="unidad" class="form-select" required>
        <option>Elige</option>
        <?php while ($u != NULL) { ?>
            <option value="<?php echo $u['id_unidad'] ?>"><?php echo $u['unidad'] ?></option>
            <?php
            $u = mysqli_fetch_assoc($luni);
        }
        ?>
    </select>
</div>
<div class="col-md-4">
    <label for="dnit1" class="form-label">Introduzca el DNI del primer tutor:* </label>
    <input name="dnit1" class="form-control" type="text" maxlength="9" required/>
</div>
<div class="col-md-4">
    <label for="nombret1" class="form-label">Introduzca el nombre del primer tutor:* </label>
    <input name="nombret1" class="form-control" type="text" required/>
</div>
<div class="col-md-4">
    <label for="sexot1" class="form-label">Introduzca el sexo del primer tutor:* </label>
    <select name="sexot1" class="form-select">
        <option value="H">H</option>
        <option value="M">M</option>
    </select>
</div>
<div class="col-md-6">
    <label for="apellido1t1" class="form-label">Introduzca el primer apellido del primer tutor:* </label>
    <input name="apellido1t1" class="form-control" type="text" required/>
</div>
<div class="col-md-6">
    <label for="apellido2t1" class="form-label">Introduzca el segundo apellido del primer tutor:* </label>
    <input name="apellido2t1" class="form-control" type="text" required/>
</div>
```

Figura 5.3.1.13 Código HTML formulario añadir alumno (3)

Los campos en los que sea necesario introducir una fecha, se mostrará un calendario en el que se podrá seleccionar el día, mes y año o introducirlo manualmente en su lugar correspondiente.

```
<div class="col-md-4">
    <label for="dnit2" class="form-label">Introduzca el DNI del segundo tutor:* </label>
    <input name="dnit2" class="form-control" type="text" maxlength="9" required/>
</div>
<div class="col-md-4">
    <label for="nombret2" class="form-label">Introduzca el nombre del segundo tutor:* </label>
    <input name="nombret2" class="form-control" type="text" required/>
</div>
<div class="col-md-4">
    <label for="sexot2" class="form-label">Introduzca el sexo del segundo tutor:* </label>
    <select name="sexot2" class="form-select">
        <option value="H">H</option>
        <option value="M">M</option>
    </select>
</div>
<div class="col-md-6">
    <label for="apellido1t2" class="form-label">Introduzca el primer apellido del segundo tutor:* </label>
    <input name="apellido1t2" class="form-control" type="text" required/>
</div>
<div class="col-md-6">
    <label for="apellido2t2" class="form-label">Introduzca el segundo apellido del segundo tutor:* </label>
    <input name="apellido2t2" class="form-control" type="text" required/>
</div>
<div class="col-md-3">
    <label for="fechamat" class="form-label">Introduzca el fecha de matricula:* </label>
    <input name="fechamat" class="form-control" type="date" required/>
</div>
<div class="col-md-4">
    <label for="nummats" class="form-label">Introduzca el n&uacute;mero de matriculas del curso actual:* </label>
    <input name="nummats" class="form-control" type="text" required maxlength="2"/>
</div>
<div class="col-md-5">
    <label for="numss" class="form-label">Introduzca el n&uacute;mero de la seguridad social: </label>
    <input name="numss" class="form-control" type="text"/>
</div>
<div class="col-md-12">
    <label for="observaciones" class="form-label">Introduzca las observaciones sobre la matricula: </label>
    <textarea name="observaciones" class="form-control" style="height: 200px"></textarea>
</div>
<!-- boton submit formulario -->
<div class="d-flex justify-content-end bd-highlight mb-3">
    <button type="submit" class="btn btn-primary">A&ntilde;adir</button>
</div>
```

Figura 5.3.1.14 Código HTML formulario añadir alumno (4)

El formulario también posee un botón llamado Volver a Gestión del alumnado el cual redirecciona a la página *galumno.php*.

Tras pulsar el botón *Añadir* el sistema llama al archivo *saveAddAlumno.php* que es el encargado de añadir a la base de datos. El sistema comprueba que el ID escolar no ha sido introducido anteriormente.

```
//selecciona todos los datos de la tabla alumno
$lal = mysqli_query($connect, "SELECT * FROM alumno") or die(mysqli_connect_error());
$row = mysqli_fetch_assoc($lal); //ejecuta la consulta
$found = 0;

while ($found == 0 && $row != NULL) { //comprueba si ya hay alguien con ese id esoclar
    if (strcasecmp($row['id_escolar'], $_POST['idescolar']) == 0) { //si exite en la tabla
        $found = 1; //marca found a 1
    } else { //si no existe
        $row = mysqli_fetch_assoc($lal); //coge el siguiente alumno de la base de datos
    }
}
```

Figura 5.3.1.15 Búsqueda ID escolar en base de datos

Si no se ha añadido, se crea la variable *alumno* combinando el nombre y los apellidos; si el usuario no ha introducido observaciones, se crea la variable *observaciones* con valor "No hay observaciones" y en el caso de que si se hayan introducido, la variable *observaciones* se iguala a la que hemos obtenido con POST. También se crea la variable *anyo_mat* que almacena el año de la fecha de matriculación que se obtuvo con POST.

```php
if ($found == 0) { //si no hay ningun alumno con ese id, se guarda en la base de datos
    $alumno = $_POST['nombre'] . " " . $_POST['apellido1'] . " " . $_POST['apellido2']; //une todo el nombre del alumno en una linea

    if ($_POST['observaciones'] == NULL) {//si no hay observaciones se rellena el campo con "no hay observaciones"
        $observaciones = "No hay observaciones";
    } else {
        $observaciones = $_POST['observaciones'];
    }
    $anyo_mat = explode("-", $_POST['fechamat']); //divide el string en 3, $anyo_mat[0] es el año de la matricula
```

Figura 5.3.1.16 Creación de variables

Y realiza la consulta con los datos obtenidos mediante el método POST y la ejecuta, por último redirige a *galumno.php* y manda por URL el parámetro *add* para que muestre la alerta correcta. En caso de que se haya añadido anteriormente, vuelve al formulario de añadir alumno y muestra un mensaje indicando que ese id ya pertenece a otro alumno.

```php
    $query = sprintf('INSERT INTO alumno (id_escolar, alumno, estado_matricula, dni_alumno, direccion, codigo_postal, '
            . 'localidad, fecha_nacimiento, provincia, telefono, tlf_emergencia, email, curso, num_expediente, unidad, '
            . 'apellido1, apellido2, nombre, dni_tutor1, apellido1_tutor1, apellido2_tutor1, nombre_tutor1, sexo_tutor1, '
            . 'dni_tutor2, apellido1_tutor2, apellido2_tutor2, nombre_tutor2, sexo_tutor2, localidad_nacimiento, '
            . 'anyo_matricula, num_matriculas_curso_actual, observaciones_matricula, provincia_nacimiento, pais_nacimiento,'
            . 'edad_final_anyo_matricula, nacionalidad, sexo, fecha_matricula, num_seg_social) VALUES ("%s", "%s", "%s", '
            . '"%s", "%s", "%s","%s", "%s", "%s", "%s", "%s", "%s", "%s", "%s","%s", "%s", "%s", "%s", "%s", '
            . '"%s", "%s", "%s", "%s", "%s","%s", "%s", "%s", "%s", "%s", "%s", "%s", "%s", "%s","%s", "%s", "%s")',
            $_POST['idescolar'], $alumno, $_POST['estadomat'], $_POST['dni'], $_POST['direccion'], $_POST['codpostal'],
            $_POST['localidad'], $_POST['fechanac'], $_POST['provincia'], $_POST['telefono'], $_POST['tlfsos'],
            $_POST['email'], $_POST['curso'], $_POST['numexp'], $_POST['unidad'], $_POST['apellido1'], $_POST['apellido2'],
            $_POST['nombre'], $_POST['dnit1'], $_POST['apellido1t1'], $_POST['apellido2t1'], $_POST['nombret1'],
            $_POST['sexot1'], $_POST['dnit2'], $_POST['apellido1t2'], $_POST['apellido2t2'], $_POST['nombret2'],
            $_POST['sexot2'], $_POST['localidadnac'], $anyo_mat[0], $_POST['nummats'], $observaciones, $_POST['provincianac'],
            $_POST['paisnac'], NULL, $_POST['nacionalidad'], $_POST['sexo'], $_POST['fechamat'], $_POST['numss']);

    echo $res = mysqli_query($connect, $query) or die(mysqli_connect_error() . 'error al insertar el alumno'); //ejecuta la consulta
    $urlr = $url."galumno.php?add=1"."&id_login=" . $_GET['id_login']; //url a la que redireccionar
    header('Location: ' . $urlr); //redirecciona a la url anterior
    exit();
} else { //si ya hay alguien con ese id, vuelve a la pagina de añadir alumno
    //vuelve al formulario de añadir
    echo '<script> alert("Ya se ha añadido un alumno con este ID escolar."); history.back();</script>';
}
```

Figura 5.3.1.17 Consulta INSERT alumno

A continuación se muestra el formulario tal y como lo vería el usuario:

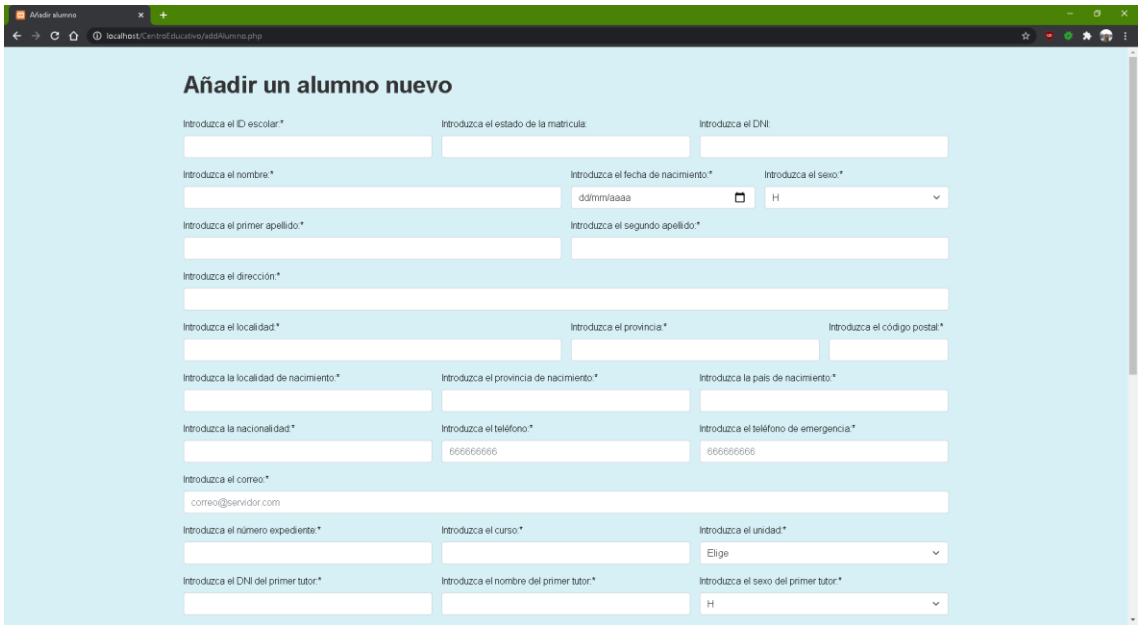

Figura 5.3.1.18 Formulario añadir alumno (1)

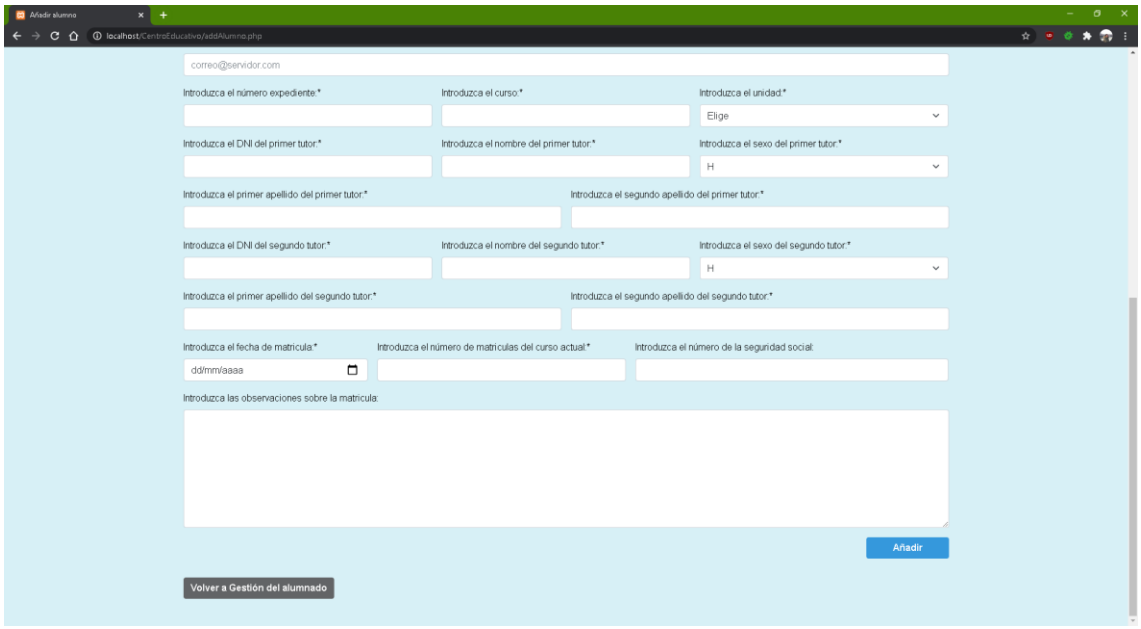

Figura 5.3.1.19 Formulario añadir alumno (2)

Tras el botón de Añadir nuevo alumno, aparece la tabla de datos. Para rellenar esta tabla, el sistema realiza una consulta SELECT que obtiene la lista de todos los alumnos que se han introducido anteriormente. En caso de que la consulta no devuelva nada, la tabla solo mostrará la cabecera.

```php
<?php
//obtiene todos los alumnos de la base de datos
$lAlum = mysqli_query($connect, "SELECT * FROM alumno") or die(mysqli_connect_error());
$row = mysqli_fetch_assoc($lAlum);
?>
```

Figura 5.3.1.20 Consulta SELECT alumnos

Una vez tenemos la lista, por cada unidad se mostrará una fila con el ID escolar del alumno, el nombre y la unidad a la que pertenece, además de cuatro opciones que se detallarán más adelante.

```
<table>
    <thead>
        <tr>
            <th>ID Escolar</th>
            <th>Nombre</th>
            <th>Unidad</th>
            <th></th>
            <th></th>
            <th>Tutoría</th>
            <th></th>
        </tr>
    </thead>
    <tbody>
        <?php
        while ($row != NULL) {
            ?>
            <tr>
                <td><?php echo $row['id_escolar'] ?></td>
                <td><?php echo $row['alumno'] ?></td>
                <td><?php echo $row['unidad'] ?></td>
```

Figura 5.3.1.21 Código HTML tabla de datos

Estas tres opciones, mencionadas anteriormente, que hacen uso de los datos son *Editar*, *Consultar, Informes* y *Eliminar*. Cada opción es un botón que, o bien te lleva a un formulario o bien realiza la acción que su propio nombre indica. Al igual que en el resto de apartados de gestión, solo estará disponible el botón consultar para todos los usuarios que inicien sesión, el resto solo lo estarán para usuarios con rol admin.

```
<?php if (strpos($_SESSION['rol'], "1") !== false) { ?>
    <td><a class="btn btn-primary" href='<?php echo $url; ?>editAlumno.php?idescolar=<?php echo $row['id_escolar'] ?>&id_login=<?php echo $_GET['id_login']; ?>'>Editar</a></td>
<?php } else { ?>
    <td><button class="btn btn-primary" disabled>Editar</button></td>
<?php } ?>
<td><a class="btn btn-primary" href= '<?php echo $url; ?>showAlumno.php?idescolar=<?php echo $row['id_escolar']; ?>&id_login=<?php echo $_GET['id_login']; ?>'>Consultar</a></td>
<td><a class="btn btn-primary" href= '<?php echo $url; ?>gInformeAlumno.php?idescolar=<?php echo $row['id_escolar']; ?>&id_login=<?php echo $_GET['id_login']; ?>'>Informes</a></td>
<?php if (strpos($_SESSION['rol'], "1") !== false) { ?>
    <td><button class="delete btn btn-danger" id="<?= $row['id_escolar'] ?>" data-id="<?= $row['id_escolar'] ?>" >Eliminar</button></td>
<?php } else { ?>
    <td><button class="delete btn btn-danger" id="<?= $row['id_escolar'] ?>" data-id="<?= $row['id_escolar'] ?>" disabled>Eliminar</button></td>
<?php } ?>
```

Figura 5.3.1.22 Código HTML opciones de tabla

Si el usuario pulsa el botón Editar, se muestra la página *editAlumno.php* a la cual se le ha enviado la *id_escolar* del alumno que se quiere editar a través de la URL y el sistema realizará la consulta SELECT para obtener los datos. También se obtendrá un listado de unidades para que el usuario pueda utilizar la lista desplegable en el caso de que quiera cambiar de unidad al alumno.

```php
$id_escolar = $_GET['idescolar']; //coge el id_escolar del alumno de la url
//obtiene todos los datos del alumno de la base de datos
$query = mysqli_query($connect, 'SELECT * FROM alumno WHERE id_escolar="' . $id_escolar . '"') or die(mysqli_connect_error() . "no coge el dni");
$alum = mysqli_fetch_assoc($query);
$luni = mysqli_query($connect, "SELECT * FROM unidad") or die(mysqli_connect_error()); //obtiene la asignatura del informe
$u = mysqli_fetch_assoc($luni);
```

Figura 5.3.1.23 Consulta SELECT de un alumno

Una vez que tenemos los datos del alumno, se realiza un formulario el cual ya está relleno.

```php
<form id="form" class=" row g-3" action="lib/saveEditAlumno.php?id_escolar=<?= $alum['id_escolar']; ?>&id_login=<?=$_GET['id_login']?>" method="POST" enctype="multipart/form-data">
    <div class="col-md-4">
        <label for="idescolar" class="form-label">Introduzca el ID escolar:* </label>
        <input name="idescolar" class="form-control" type="text" readonly disable value="<?= $alum['id_escolar']; ?>"/>
    </div>
    <div class="col-md-4">
        <label for="estadomat" class="form-label">Introduzca el estado de la matricula: </label>
        <input name="estadomat" class="form-control" type="text" value="<?= $alum['estado_matricula']; ?>"/>
    </div>
    <div class="col-md-4">
        <label for="dni" class="form-label">Introduzca el DNI: </label>
        <input name="dni" class="form-control" type="text" maxlength="9" value="<?= $alum['dni_alumno']; ?>"/>
    </div>
    <div class="col-md-6">
        <label for="nombre" class="form-label">Introduzca el nombre:* </label>
        <input name="nombre" class="form-control" type="text" required value="<?= $alum['nombre']; ?>"/>
    </div>
    <div class="col-md-3">
        <label for="fechanac" class="form-label">Introduzca el fecha de nacimiento:* </label>
        <input name="fechanac" class="form-control" type="date" required value="<?= $alum['fecha_nacimiento']; ?>"/>
    </div>
    <div class="col-md-3">
        <label for="sexo" class="form-label">Introduzca el sexo:* </label>
        <select name="sexo" class="form-select" required>
            <?php if (strcmp($alum['sexo'], "H") == 0) { ?>
                <option value="H" selected>H</option>
                <option value="M">M</option>
            <?php } else { ?>
                <option value="H">H</option>
                <option value="M" selected>M</option>
            <?php } ?>
        </select>
    </div>
    <div class="col-md-6">
        <label for="apellido1" class="form-label">Introduzca el primer apellido:* </label>
        <input name="apellido1" class="form-control" type="text" required value="<?= $alum['apellido1']; ?>"/>
    </div>
    <div class="col-md-6">
        <label for="apellido2" class="form-label">Introduzca el segundo apellido:* </label>
        <input name="apellido2" class="form-control" type="text" required value="<?= $alum['apellido2']; ?>"/>
    </div>
```

Figura 5.3.1.24 Código HTML formulario editar alumno (1)

El formulario tendrá seleccionado tanto en los campos *Sexo*, como en el campo *unidad*, los datos que se encuentran en la base de datos, es decir, los últimos que el usuario introdujo o editó. Estos campos siguen teniendo disponible una lista desplegable para poder cambiar el valor.

Los campos relacionados con los teléfonos y el correo siguen teniendo las mismas restricciones que en el formulario de añadir nuevo alumno.

```
<div class="col-md-12">
    <label for="direccion" class="form-label">Introduzca el direcci&oacute;n:* </label>
    <input name="direccion" class="form-control" type="text" required value="<?= $alum['direccion']; ?>"/>
</div>
<div class="col-md-6">
    <label for="localidad" class="form-label">Introduzca el localidad:* </label>
    <input name="localidad" class="form-control" type="text" required value="<?= $alum['localidad']; ?>"/>
</div>
<div class="col-md-4">
    <label for="provincia" class="form-label">Introduzca el provincia:* </label>
    <input name="provincia" class="form-control" type="text" required value="<?= $alum['provincia']; ?>"/>
</div>
<div class="col-md-2">
    <label for="codpostal" class="form-label">Introduzca el c&oacute;digo postal:* </label>
    <input name="codpostal" class="form-control" type="text" required value="<?= $alum['codigo_postal']; ?>"/>
</div>
<div class="col-md-4">
    <label for="localidadnac" class="form-label">Introduzca la localidad de nacimiento:* </label>
    <input name="localidadnac" class="form-control" type="text" required value="<?= $alum['localidad_nacimiento']; ?>"/>
</div>
<div class="col-md-4">
    <label for="provincianac" class="form-label">Introduzca el provincia de nacimiento:* </label>
    <input name="provincianac" class="form-control" type="text" required value="<?= $alum['provincia_nacimiento']; ?>"/>
</div>
<div class="col-md-4">
    <label for="paisnac" class="form-label">Introduzca la pa&iacute;s de nacimiento:* </label>
    <input name="paisnac" class="form-control" type="text" required value="<?= $alum['pais_nacimiento']; ?>"/>
</div>
<div class="col-md-4">
    <label for="nacionalidad" class="form-label">Introduzca la nacionalidad:* </label>
    <input name="nacionalidad" class="form-control" type="text" required value="<?= $alum['nacionalidad']; ?>"/>
</div>
<div class="col-md-4">
    <label for="telefono" class="form-label">Introduzca el tel&eacute;fono:* </label>
    <input name="telefono" class="form-control" type="tel" placeholder="666666666" required value="<?= $alum['telefono']; ?>"/>
</div>
<div class="col-md-4">
    <label for="tlfsos" class="form-label">Introduzca el tel&eacute;fono de emergencia:* </label>
    <input name="tlfsos" class="form-control" type="tel" placeholder="666666666" required value="<?= $alum['tlf_emergencia']; ?>"/>
</div>
```

Figura 5.3.1.25 Código HTML formulario editar alumno (2)

```
<div class="col-md-12">
    <label for="email" class="form-label">Introduzca el correo:* </label>
    <input name="email" class="form-control" type="email" placeholder="correo@servidor.com" required value="<?= $alum['email']; ?>"/>
</div>
<div class="col-md-4">
    <label for="numexp" class="form-label">Introduzca el n&uacute;mero expediente:* </label>
    <input name="numexp" class="form-control" type="text" value="<?= $alum['num_expediente']; ?>"/>
</div>
<div class="col-md-4">
    <label for="curso" class="form-label">Introduzca el curso:* </label>
    <input name="curso" class="form-control" type="text" required value="<?= $alum['curso']; ?>"/>
</div>
<div class="col-md-4">
    <label for="unidad" class="form-label">Introduzca el unidad:* </label>
    <select name="unidad" class="form-select" required>
        <option>Elige</option>
        <?php
        while ($u != NULL) {
            if (strcmp($alum['unidad'], $u['unidad']) == 0) {
                ?>
                <option selected value = "<?php echo $u['unidad'] ?>"><?php echo $u['unidad'] ?></option>
                <?php
            } else {
                ?>
                <option value="<?php echo $u['unidad'] ?>"><?php echo $u['unidad'] ?></option>
                <?php
            }
            $u = mysqli_fetch_assoc($luni);
        }
        ?>
    </select>
</div>
<div class="col-md-4">
    <label for="dnit1" class="form-label">Introduzca el DNI del primer tutor:* </label>
    <input name="dnit1" class="form-control" type="text" readonly disable value="<?= $alum['dni_tutor1']; ?>"/>
</div>
<div class="col-md-4">
    <label for="nombret1" class="form-label">Introduzca el nombre del primer tutor:* </label>
    <input name="nombret1" class="form-control" type="text" required value="<?= $alum['nombre_tutor1']; ?>"/>
</div>
```

Figura 5.3.1.26 Código HTML formulario editar alumno (3)

```
<div class="col-md-4">
    <label for="sexot1" class="form-label">Introduzca el sexo del primer tutor:* </label>
    <select name="sexot1" class="form-select" required>
        <?php if (strcmp($alum['sexo_tutor1'], "H") == 0) { ?>
            <option value="H" selected>H</option>
            <option value="M">M</option>
        <?php } else { ?>
            <option value="H">H</option>
            <option value="M" selected>M</option>
        <?php } ?>
    </select>
</div>
<div class="col-md-6">
    <label for="apellido1t1" class="form-label">Introduzca el primer apellido del primer tutor:* </label>
    <input name="apellido1t1" class="form-control" type="text" required value="<?= $alum['apellido1_tutor1']; ?>"/>
</div>
<div class="col-md-6">
    <label for="apellido2t1" class="form-label">Introduzca el segundo apellido del primer tutor:* </label>
    <input name="apellido2t1" class="form-control" type="text" required value="<?= $alum['apellido2_tutor1']; ?>"/>
</div>
<div class="col-md-4">
    <label for="dnit2" class="form-label">Introduzca el DNI del segundo tutor:* </label>
    <input name="dnit2" class="form-control" type="text" readonly disable value="<?= $alum['dni_tutor2']; ?>"/>
</div>
<div class="col-md-4">
    <label for="nombret2" class="form-label">Introduzca el nombre del segundo tutor:* </label>
    <input name="nombret2" class="form-control" type="text" required value="<?= $alum['nombre_tutor2']; ?>"/>
</div>
<div class="col-md-4">
    <label for="sexot2" class="form-label">Introduzca el sexo del segundo tutor:* </label>
    <select name="sexot2" class="form-select" required>
        <?php if (strcmp($alum['sexo_tutor2'], "H") == 0) { ?>
            <option value="H" selected>H</option>
            <option value="M">M</option>
        <?php } else { ?>
            <option value="H">H</option>
            <option value="M" selected>M</option>
        <?php } ?>
    </select>
</div>
```

Figura 5.3.1.27 Código HTML formulario editar alumno (4)

En los campos que contengan fechas, aparecerá la última fecha que se introdujo o editó, además del calendario antes mencionado para poder cambiar si el usuario lo desea.

```
    <div class="col-md-6">
        <label for="apellido1t2" class="form-label">Introduzca el primer apellido del segundo tutor:* </label>
        <input name="apellido1t2" class="form-control" type="text" required value="<?= $alum['apellido1_tutor2']; ?>"/>
    </div>
    <div class="col-md-6">
        <label for="apellido2t2" class="form-label">Introduzca el segundo apellido del segundo tutor:* </label>
        <input name="apellido2t2" class="form-control" type="text" required value="<?= $alum['apellido2_tutor2']; ?>"/>
    </div>
    <div class="col-md-3">
        <label for="fechamat" class="form-label">Introduzca el fecha de matricula:* </label>
        <input name="fechamat" class="form-control" type="date" value="<?= $alum['fecha_matricula']; ?>"/>
    </div>
    <div class="col-md-4">
        <label for="nummats" class="form-label">Introduzca el n&uacute;mero de matriculas del curso actual:* </label>
        <input name="nummats" class="form-control" type="text" required maxlength="2" value="<?= $alum['num_matriculas_curso_actual']; ?>"/>
    </div>
    <div class="col-md-5">
        <label for="numss" class="form-label">Introduzca el n&uacute;mero de la seguridad social: </label>
        <input name="numss" class="form-control" type="text" value="<?= $alum['num_seg_social']; ?>"/>
    </div>
    <div class="col-md-12">
        <label for="observaciones" class="form-label">Introduzca las observaciones sobre la matricula: </label>
        <textarea name="observaciones" class="form-control" style="height: 200px" value="<?= $alum['observaciones_matricula']; ?>"><?= $alum['observaciones_matricula']; ?></textarea>
    </div>
    <!-- boton submint formulario -->
    <div class="d-flex justify-content-end bd-highlight mb-3">
        <button id="editar" type="submit" class="btn btn-primary" disabled>Editar</button>
    </div>
</form>
```

Figura 5.3.1.28 Código HTML formulario editar alumno (5)

En el momento en el que el usuario modifique algún dato, el botón llamado *Editar* que se encuentra al final del formulario se habilitará y cuando sea pulsado, el sistema llamará al archivo *saveEditAlumno.php* que realizará la consulta UPDATE con los parámetros que se obtuvieron utilizando POST y actualizará los datos del alumno. Para saber qué unidad tenemos que actualizar, se pasará por URL la *id escolar* del alumno. El sistema recogerá la id y se crearán las variables necesarias, al igual que en añadir alumno.

```php
$id_escolar = $_GET['id_escolar']; //coge el id de la url
//une todo el nombre del alumno en una linea
$alumno = $_POST['nombre'] . " " . $_POST['apellido1'] . " " . $_POST['apellido2'];
$anyo_mat = explode("-", $_POST['fechamat']); //divide el string en 3, $anyo_mat[0] es el año de la matricula

if ($_POST['observaciones'] == NULL) { //si no hay observaciones se rellena el campo con "no hay observaciones"
    $observaciones = "No hay observaciones";
} else {
    $observaciones = $_POST['observaciones'];
}
```

Figura 5.3.1.29 Creación de variables

Una vez está todo creado, se actualizarán la base de datos, el sistema volverá a la página principal de la gestión de unidades y se mostrará un mensaje diciendo que se han editado los datos correctamente.

```php
$query = "UPDATE alumno SET alumno='" . $alumno . "', estado_matricula='" . $_POST['estadomat'] . "', dni_alumno='"
        . $_POST['dni'] . "', direccion='" . $_POST['direccion'] . "', codigo_postal='" . $_POST['codpostal']
        . "', localidad='" . $_POST['localidad'] . "', fecha_nacimiento='" . $_POST['fechanac'] . "', "
        . "provincia='" . $_POST['provincia'] . "', telefono='" . $_POST['telefono'] . "', tlf_emergencia='"
        . $_POST['tlfsos'] . "', email='" . $_POST['email'] . "', curso='" . $_POST['curso'] . "', num_expediente='"
        . $_POST['numexp'] . "', unidad='" . $_POST['unidad'] . "', apellido1='" . $_POST['apellido1'] . "', apellido2='"
        . $_POST['apellido2'] . "', nombre='" . $_POST['nombre'] . "', dni_tutor1='" . $_POST['dnit1'] . "', apellido1_tutor1='"
        . $_POST['apellido1t1'] . "', apellido2_tutor1='" . $_POST['apellido2t1'] . "', nombre_tutor1='" . $_POST['nombret1']
        . "', sexo_tutor1='" . $_POST['sexot1'] . "', dni_tutor2='" . $_POST['dnit2'] . "', apellido1_tutor2='" . $_POST['apellido1t2']
        . "', apellido2_tutor2='" . $_POST['apellido2t2'] . "', nombre_tutor2='" . $_POST['nombret2'] . "', sexo_tutor2='"
        . $_POST['sexot2'] . "', localidad_nacimiento='" . $_POST['localidadnac'] . "', anyo_matricula='" . $anyo_mat[0]
        . "', num_matriculas_curso_actual='" . $_POST['nummats'] . "', observaciones_matricula='" . $observaciones . "', "
        . "provincia_nacimiento='" . $_POST['provincianac'] . "', pais_nacimiento='" . $_POST['paisnac']
        . "', edad_final_anyo_matricula='" . NULL . "', nacionalidad='" . $_POST['nacionalidad'] . "', "
        . "sexo='" . $_POST['sexo'] . "', fecha_matricula='" . $_POST['fechamat'] . "', num_seg_social='"
        . $_POST['numss'] . "' WHERE id_escolar='" . $id_escolar . "'";

echo $res = mysqli_query($connect, $query) or die(mysqli_connect_error() . 'error al editar el alumno'); //ejecuta la consulta
$urlr = $url."galumno.php?ed=1"."&id_login=" . $_GET['id_login'];; //url a la que redireccionar
header('Location: ' . $urlr); //redirecciona a la url anterior
```

Figura 5.3.1.30 Consulta UPDATE alumno

Como el resto de formularios de esta sección, también posee un botón llamado Volver a Gestión del alumnado el cual redirecciona a la página *galumno.php*.

A continuación se mostrará un ejemplo del formulario de edición del alumno:

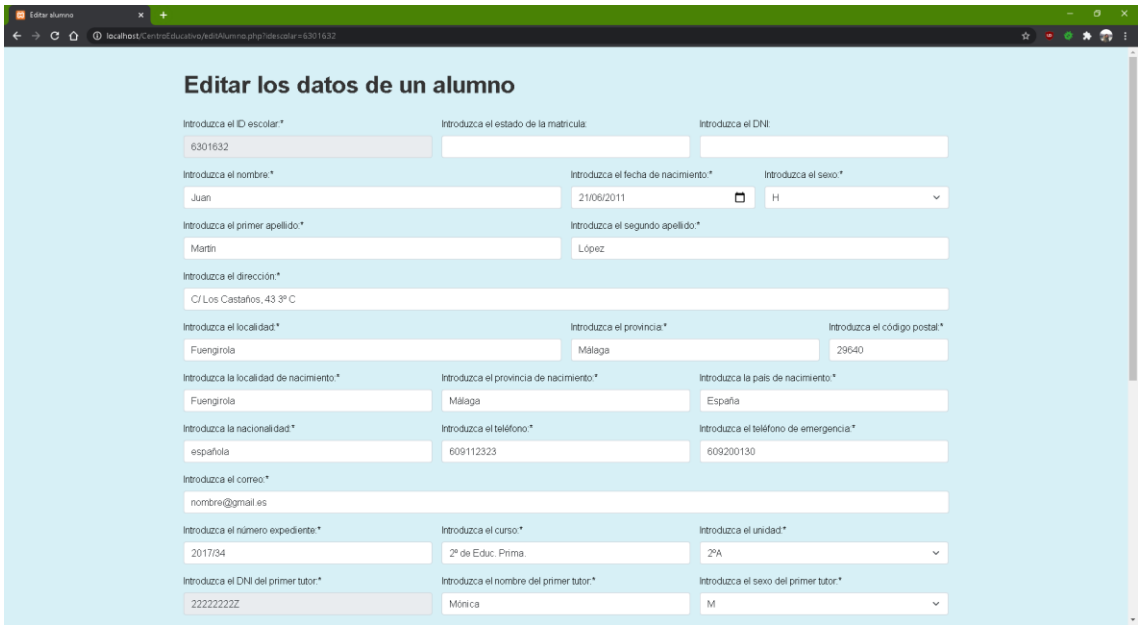

Figura 5.3.1.31 Formulario editar alumno (1)

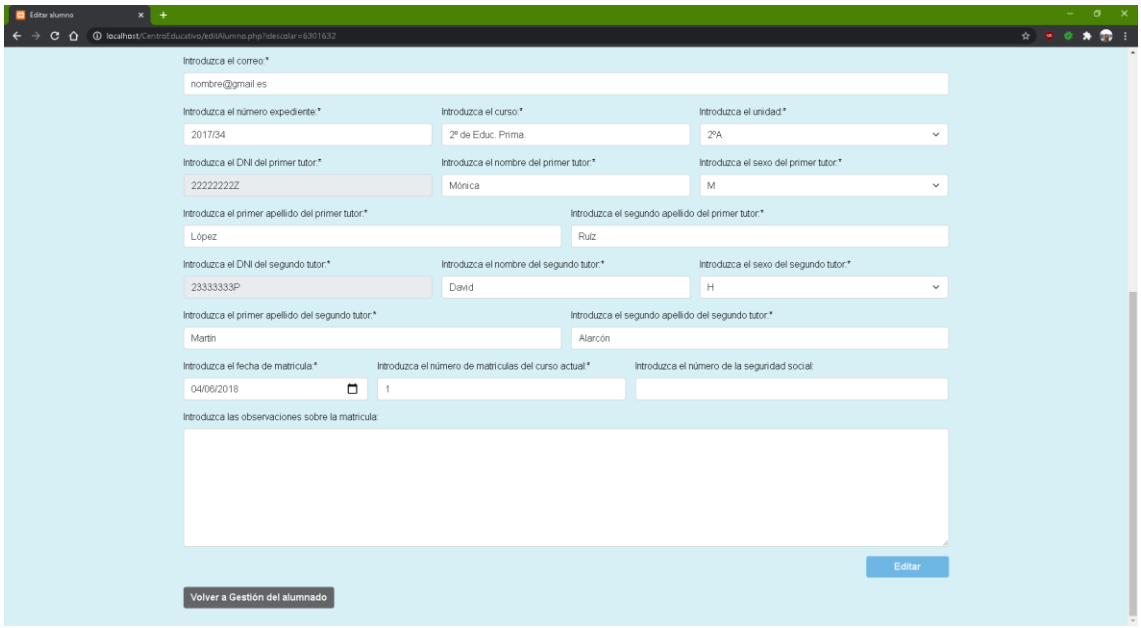

Figura 5.3.1.32 Formulario editar alumno (2)

Es la segunda opción disponible para cada fila de datos de la tabla. Cuando el usuario pulsa el botón *Consultar*, el sistema muestra la página *showAlumno.php* la cual es un formulario que muestra los datos de ese alumno en concreto. Para ello se envía por URL la *id escolar* del alumno que se desea consultar y el sistema realiza una consulta SELECT para obtener los datos de la base de datos. Estos datos no se pueden editar, solo se muestran a modo de consulta para el usuario.

```
<form id="form" class=" row g-3" enctype="multipart/form-data">
    <div class="col-md-4">
        <label for="idescolar" class="form-label">ID escolar: </label>
        <input name="idescolar" class="form-control" type="text" readonly disabled value="<?= $alum['id_escolar']; ?>"/>
    </div>
    <div class="col-md-4">
        <label for="estadomat" class="form-label">Estado de la matricula: </label>
        <input name="estadomat" class="form-control" type="text" readonly disabled value="<?= $alum['estado_matricula']; ?>"/>
    </div>
    <div class="col-md-4">
        <label for="dni" class="form-label">DNI: </label>
        <input name="dni" class="form-control" type="text" maxlength="9" readonly disabled value="<?= $alum['dni_alumno']; ?>"/>
    </div>
    <div class="col-md-6">
        <label for="nombre" class="form-label">Nombre: </label>
        <input name="nombre" class="form-control" type="text" required readonly disabled value="<?= $alum['nombre']; ?>"/>
    </div>
    <div class="col-md-3">
        <label for="fechanac" class="form-label">Fecha de nacimiento: </label>
        <input name="fechanac" class="form-control" type="date" required readonly disabled value="<?= $alum['fecha_nacimiento']; ?>"/>
    </div>
    <div class="col-md-3">
        <label for="sexo" class="form-label">Sexo: </label>
        <select name="sexo" class="form-select" required disabled >
            <?php if (strcmp($alum['sexo'], "H") == 0) { ?>
                <option value="H" selected>H</option>
                <option value="M">M</option>
            <?php } else { ?>
                <option value="H">H</option>
                <option value="M" selected>M</option>
            <?php } ?>
        </select>
    </div>
    <div class="col-md-6">
        <label for="apellido1" class="form-label">Primer apellido: </label>
        <input name="apellido1" class="form-control" type="text" required readonly disabled value="<?= $alum['apellido1']; ?>"/>
    </div>
    <div class="col-md-6">
        <label for="apellido2" class="form-label">Segundo apellido: </label>
        <input name="apellido2" class="form-control" type="text" required readonly disabled value="<?= $alum['apellido2']; ?>"/>
    </div>
```

Figura 5.3.1.33 Código HTML formulario consultar alumno (1)

```
<div class="col-md-12">
    <label for="direccion" class="form-label">Direcci&oacute;n: </label>
    <input name="direccion" class="form-control" type="text" required readonly disabled value="<?= $alum['direccion']; ?>"/>
</div>
<div class="col-md-6">
    <label for="localidad" class="form-label">Localidad: </label>
    <input name="localidad" class="form-control" type="text" required readonly disabled value="<?= $alum['localidad']; ?>"/>
</div>
<div class="col-md-4">
    <label for="provincia" class="form-label">Provincia: </label>
    <input name="provincia" class="form-control" type="text" required readonly disabled value="<?= $alum['provincia']; ?>"/>
</div>
<div class="col-md-2">
    <label for="codpostal" class="form-label">C&oacute;digo postal: </label>
    <input name="codpostal" class="form-control" type="text" required readonly disabled value="<?= $alum['codigo_postal']; ?>"/>
</div>
<div class="col-md-4">
    <label for="localidadnac" class="form-label">Localidad de nacimiento: </label>
    <input name="localidadnac" class="form-control" type="text" required readonly disabled value="<?= $alum['localidad_nacimiento']; ?>"/>
</div>
<div class="col-md-4">
    <label for="provincianac" class="form-label">Provincia de nacimiento: </label>
    <input name="provincianac" class="form-control" type="text" required readonly disabled value="<?= $alum['provincia_nacimiento']; ?>"/>
</div>
<div class="col-md-4">
    <label for="paisnac" class="form-label">Pa&iacute;s de nacimiento: </label>
    <input name="paisnac" class="form-control" type="text" required readonly disabled value="<?= $alum['pais_nacimiento']; ?>"/>
</div>
<div class="col-md-4">
    <label for="nacionalidad" class="form-label">Nacionalidad: </label>
    <input name="nacionalidad" class="form-control" type="text" required readonly disabled value="<?= $alum['nacionalidad']; ?>"/>
</div>
<div class="col-md-4">
    <label for="telefono" class="form-label">Tel&eacute;fono: </label>
    <input name="telefono" class="form-control" type="tel" placeholder="666666666" required readonly disabled value="<?= $alum['telefono']; ?>"/>
</div>
<div class="col-md-4">
    <label for="tlfsos" class="form-label">Tel&eacute;fono de emergencia: </label>
    <input name="tlfsos" class="form-control" type="tel" placeholder="666666666" required readonly disabled value="<?= $alum['tlf_emergencia']; ?>"/>
</div>
```

Figura 5.3.1.34 Código HTML formulario consultar alumno (2)

```
<div class="col-md-12">
    <label for="email" class="form-label">Correo: </label>
    <input name="email" class="form-control" type="email" placeholder="correo@servidor.com" required readonly disabled value="<?= $alum['email']; ?>"/>
</div>
<div class="col-md-4">
    <label for="numexp" class="form-label">N&uacute;mero expediente: </label>
    <input name="numexp" class="form-control" type="text" readonly disabled value="<?= $alum['num_expediente']; ?>"/>
</div>
<div class="col-md-4">
    <label for="curso" class="form-label">Curso: </label>
    <input name="curso" class="form-control" type="text" required readonly disabled value="<?= $alum['curso']; ?>"/>
</div>
<div class="col-md-4">
    <label for="unidad" class="form-label">Unidad: </label>
    <input name="unidad" class="form-control" type="text" required readonly disabled value="<?= $alum['unidad']; ?>"/>
</div>
<div class="col-md-4">
    <label for="dnit1" class="form-label">DNI del primer tutor: </label>
    <input name="dnit1" class="form-control" type="text" readonly disabled value="<?= $alum['dni_tutor1']; ?>"/>
</div>
<div class="col-md-4">
    <label for="nombret1" class="form-label">Nombre del primer tutor: </label>
    <input name="nombret1" class="form-control" type="text" required readonly disabled value="<?= $alum['nombre_tutor1']; ?>"/>
</div>
<div class="col-md-4">
    <label for="sexot1" class="form-label">Sexo del primer tutor: </label>
    <select name="sexot1" class="form-select" requiered disabled >
        <?php if (strcmp($alum['sexo_tutor1'], "H") == 0) { ?>
            <option value="H" selected>H</option>
            <option value="M">M</option>
        <?php } else { ?>
            <option value="H">H</option>
            <option value="M" selected>M</option>
        <?php } ?>
    </select>
</div>
<div class="col-md-6">
    <label for="apellido1t1" class="form-label">Primer apellido del primer tutor: </label>
    <input name="apellido1t1" class="form-control" type="text" required readonly disabled value="<?= $alum['apellido1_tutor1']; ?>"/>
</div>
```

Figura 5.3.1.35 Código HTML formulario consultar alumno (3)

```
<div class="col-md-6">
    <label for="apellido2t1" class="form-label">Segundo apellido del primer tutor: </label>
    <input name="apellido2t1" class="form-control" type="text" required readonly disabled value="<?= $alum['apellido2_tutor1']; ?>"/>
</div>
<div class="col-md-4">
    <label for="dnit2" class="form-label">DNI del segundo tutor: </label>
    <input name="dnit2" class="form-control" type="text" readonly disabled value="<?= $alum['dni_tutor2']; ?>"/>
</div>
<div class="col-md-4">
    <label for="nombret2" class="form-label">Nombre del segundo tutor: </label>
    <input name="nombret2" class="form-control" type="text" required readonly disabled value="<?= $alum['nombre_tutor2']; ?>"/>
</div>
<div class="col-md-4">
    <label for="sexot2" class="form-label">Sexo del segundo tutor: </label>
    <select name="sexot2" class="form-select" required readonly disabled >
        <?php if (strcmp($alum['sexo_tutor2'], "H") == 0) { ?>
            <option value="H" selected>H</option>
            <option value="M">M</option>
        <?php } else { ?>
            <option value="H">H</option>
            <option value="M" selected>M</option>
        <?php } ?>
    </select>
</div>
<div class="col-md-6">
    <label for="apellido1t2" class="form-label">Primer apellido del segundo tutor: </label>
    <input name="apellido1t2" class="form-control" type="text" required readonly disabled value="<?= $alum['apellido1_tutor2']; ?>"/>
</div>
<div class="col-md-6">
    <label for="apellido2t2" class="form-label">Segundo apellido del segundo tutor: </label>
    <input name="apellido2t2" class="form-control" type="text" required readonly disabled value="<?= $alum['apellido2_tutor2']; ?>"/>
</div>
```

Figura 5.3.1.36 Código HTML formulario consultar alumno (4)

```
<div class="col-md-3">
    <label for="fechamat" class="form-label">Fecha de matricula: </label>
    <input name="fechamat" class="form-control" type="date" readonly disable readonly disabled value="<?= $alum['fecha_matricula']; ?>"/>
</div>
<div class="col-md-4">
    <label for="nummats" class="form-label">N&uacute;mero de matriculas del curso actual: </label>
    <input name="nummats" class="form-control" type="text" required maxlength="2" readonly disabled value="<?= $alum['num_matriculas_curso_actual']; ?>"/>
</div>
<div class="col-md-5">
    <label for="numss" class="form-label">N&uacute;mero de la seguridad social: </label>
    <input name="numss" class="form-control" type="text" readonly disabled value="<?= $alum['num_seg_social']; ?>"/>
</div>
<div class="col-md-12">
    <label for="observaciones" class="form-label">Observaciones sobre la matricula: </label>
    <textarea name="observaciones" class="form-control" style="height: 200px" readonly disabled value="<?= $alum['observaciones_matricula']; ?>"></textarea>
</div>
```

Figura 5.3.1.37 Código HTML formulario consultar alumno (5)

Posee un botón llamado Volver a Gestión del (*galumno.php*). A continuación se mostrará un ejemplo del formulario de consulta del alumno:

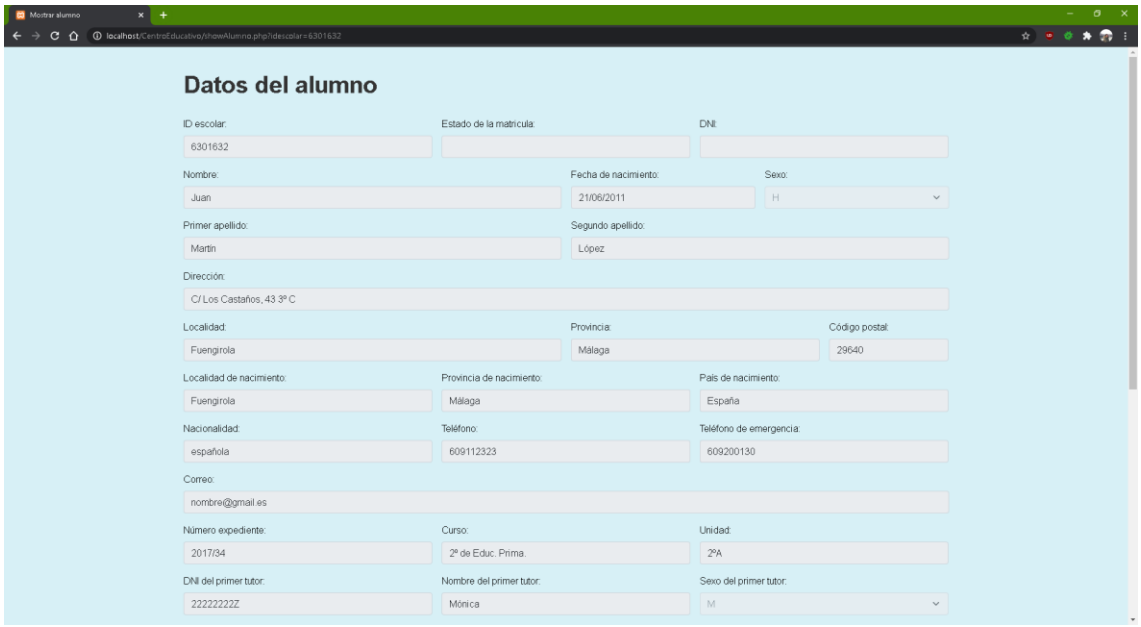

Figura 5.3.1.38 Formulario consultar alumno (1)

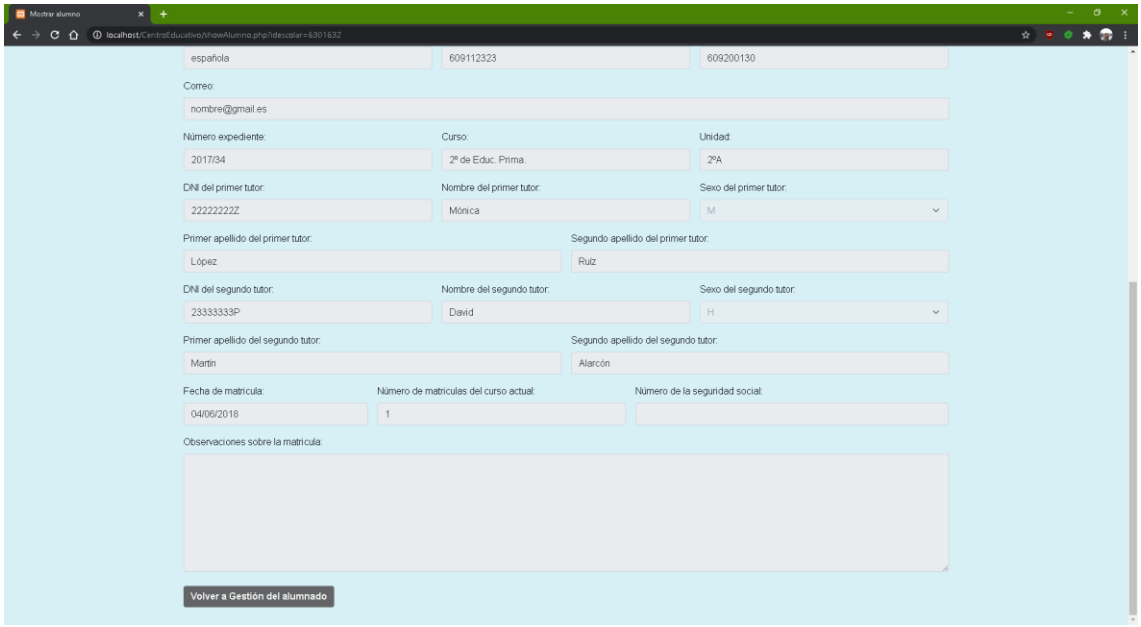

Figura 5.3.1.39 Formulario consultar alumno (2)

La tercera opción es un botón que envía al usuario a los informes de tutoría que pertenecen al alumno en cuya fila se ha pulsado el botón *Informes*, es decir, los informes que han rellenado los profesores sobre ese alumno, sin importar la asignatura. Esta opción se describirá en el apartado 5.3.2.

La última acción a realizar sobre cada unidad es la de eliminar. El usuario pulsará el botón *Eliminar* y el sistema ejecutará el siguiente código JavaScript que lo que hace es mostrar una mensaje preguntando si el usuario realmente desea eliminar el registro. En caso de que el usuario cierre el mensaje o haga click en la opción *Cancel*, se cerrará el mensaje y el sistema no eliminará nada.

```
// funcion para borrar filas de la tabla
$(document).ready(function () {
    $('.delete').click(function () {
        var el = this;
        //coge la id del elemento a borrar
        var deleteid = $(this).data('id');
        // muestra caja de confirmación
        bootbox.confirm("¿Realmente quiere eliminar?", function (result) {
            if (result) {
                //borra el elemento y actualiza la tabla con ajax
                $.ajax({
                    url: 'lib/deleteAlumno.php',
                    type: 'POST',
                    data: {id: deleteid},
```

Figura 5.3.1.40 Código JavaScript eliminar alumno (1)

En caso de que el usuario haga click en la opción *Ok*, el sistema ejecutará el archivo *deleteAlumno.php*. En este archivo, el sistema obtiene la *id escolar* del alumno que se quiere eliminar, se ejecuta la consulta DELETE y muestra el número 1 en caso de que se haya podido eliminar. Si no se pudo eliminar, el sistema mostrará el número 0.

```
if (isset($_POST['id'])) { //si el id existe y no es null
    $id_escolar = $_POST['id']; //obtiene el id del formulario

    $query = "DELETE FROM alumno WHERE id_escolar='" . $id_escolar . "'"; //consulta
    $res = mysqli_query($connect, $query) or die(mysqli_connect_error() . 'error al eliminar el alumno');
    echo 1;
    exit;
} else {
    echo 0;
    exit;
}
```

Figura 5.3.1.41 Código PHP eliminar alumno

Si el sistema recibe el número 1 como respuesta, la fila que se ha eliminado de la base de datos se mostrará en rojo y desaparecerá de la tabla usando AJAX. En caso de que la respuesta haya sido 0, se mostrará un mensaje diciendo que no se pudo eliminar el registro.

```
success: function (response) {
    //borra el elemento
    if (response == 1) {
        $(el).closest('tr').css('background', 'tomato');
        $(el).closest('tr').fadeOut(800, function () {
            $(this).remove();
        });
    } else { // si no se ha podido eleminar muestra el mensaje
        bootbox.alert('No se ha eliminado.');
    }
}
```

Figura 5.3.1.42 Código JavaScript eliminar alumno (2)

Para terminar este apartado, justo debajo de la tabla hay un botón para volver a la página de inicio, en el caso de que el usuario quiera acceder a otro apartado de gestión.

5.3.2 Gestión de informes de tutoría

La página principal de la sección de Gestión de informes de tutoría para cada alumno es la siguiente:

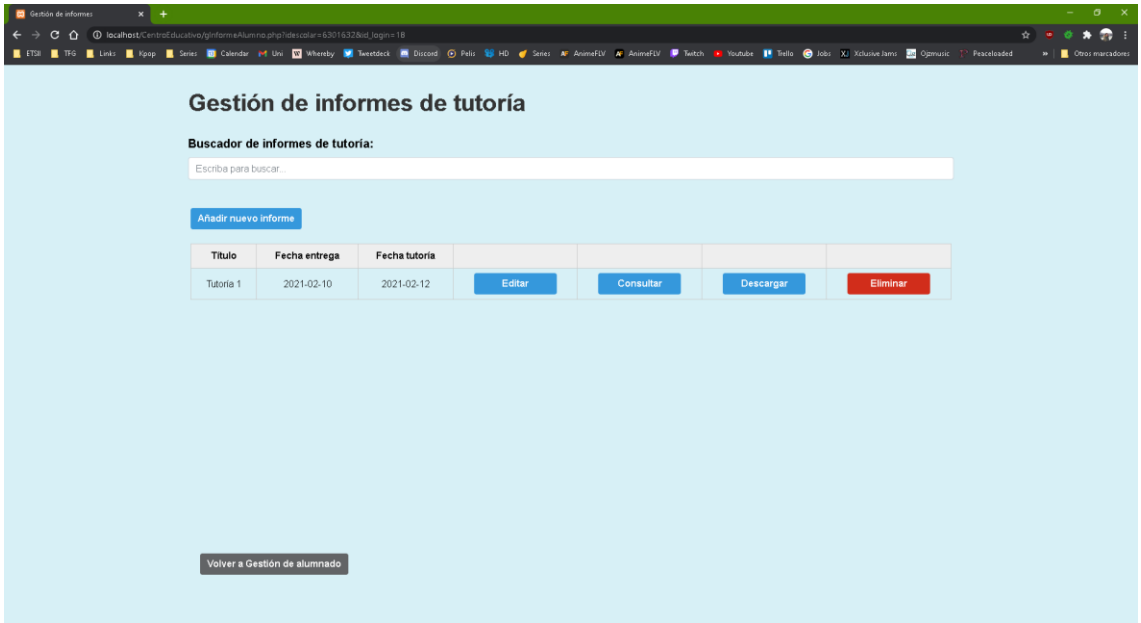

Figura 5.3.2.1 Página principal de informes de tutoría

En cuanto carga la página, el sistema obtiene todos los datos de la tabla login del usuario que ha iniciado sesión en la aplicación y si su rol es el de profesor (2) o tutor (3), obtiene los datos del profesor que coincide con el usuario que ha iniciado sesión y el listado de profesores que dan clase al alumno cuyo informe se va a realizar.

```php
//obtiene los datos de quien ha iniciado sesión
$ql = mysqli_query($connect, "SELECT * FROM login WHERE id_login='" . $_GET['id_login'] . "'");
$l = mysqli_fetch_assoc($ql);
$foundp = 0; //se pone a 1 si el profesor coincide con quien ha iniciado sesión
$found = 0; //se pone a 1 si el tutor del alumno coincide con la persona que inició sesion
if (strpos($l['rol'], "2") !== false || strpos($l['rol'], "3") !== false) {
    //obtiene los datos como profesor de quien ha hecho login
    $qp = mysqli_query($connect, "SELECT * FROM profesor WHERE email='" . $l['email'] . "'");
    $p = mysqli_fetch_assoc($qp);
    //obtiene los datos del alumno del informe
    $qa = mysqli_query($connect, "SELECT * FROM alumno WHERE id_escolar='" . $_GET['idescolar'] . "'");
    $a = mysqli_fetch_assoc($qa);
    //obtiene los datos de los profesores que dan clase y coincide con el que ha iniciado sesión
    $qm = mysqli_query($connect, "SELECT * FROM materia WHERE unidad='" . $a['unidad'] . "'");
    $m = mysqli_fetch_assoc($qm);
```

Figura 5.3.2.2 Obtención de datos de la BD

A continuación, se comprueba que el profesor que ha iniciado sesión es uno de los profesores que da clase al alumno, en tal caso se pondrá la variable *foundp* a 1. También se comprueba que el profesor que ha iniciado sesión es el tutor de la unidad a la que pertenece el alumno, si lo es se pondrá la variable *found* a 1. Estas dos variables se utilizarán más adelante para restringir el acceso a ciertas acciones de la aplicación.

```php
while ($foundp == 0 && $m != NULL) {
    if (strcasecmp($m['profesor'], $p['profesor']) == 0) {
        $foundp = 1;
    } else {
        $m = mysqli_fetch_assoc($qm);
    }
}
//obtiene los datos del tutor de la unidad y si coincide con el que ha iniciado sesión
$qu = mysqli_query($connect, "SELECT * FROM unidad WHERE unidad='" . $a['unidad'] . "'");
$u = mysqli_fetch_assoc($qu);
while ($found == 0 && $u != NULL) {
    if (strpos($u['tutor'], $p['profesor']) !== false) {
        $found = 1;
    } else {
        $u = mysqli_fetch_assoc($qu);
    }
}
```

Figura 5.3.2.3 Comprobación de la identidad del usuario

En esta página, lo primero que encontramos es el buscador en el cual se puede buscar por título, profesor o asignatura, es decir, si se quiere buscar todos los informes de la tutoría 1 solo es necesario poner en el buscador *tutoría 1* y en la tabla aparecería la lista. Este buscador, va actualizando la lista de la tabla conforme el usuario va escribiendo, por lo que no es necesario pulsar ningún botón para buscar el texto introducido.

```html
<!-- buscar un informe en la base de datos -->
<div class="container">
    <label for="buscador" class="form-label">Buscador de informes de tutor&iacute;a: </label>
    <input class="form-control" id="buscador" type="text" autocomplete="off" placeholder="Escriba para buscar..." /><br>
</div><br>
```

Figura 5.3.2.4 Código HTML del buscador

La función que realiza la acción de buscar lo que hace es comprobar si alguna fila tiene algún dato que coincida con lo buscado, en cuyo caso la muestra, si no coincide la esconde. Es la misma función que utiliza el resto de apartados de la web.

Cada vez que se cree o edite un informe, el sistema mostrará una alerta verde indicando que se ha creado o editado correctamente. Según el parámetro que se reciba por URL y el valor del parámetro, el sistema mostrará una alerta u otra. Si recibe el parámetro *add* con valor 1 significa que se han añadido los datos correctamente y si es *ed* con valor 1 es que se han editado los datos correctamente.

```php
<?php
if (isset($_GET['ok']) && $_GET['ok'] == 1) {
    echo '<div class="container alert alert-success" role="alert">Se ha añadido el informe correctamente</div>';
}
if(isset($_GET['ed']) && $_GET['ed'] == 0){
    echo '<div class="container alert alert-success" role="alert">Se ha editado el informe correctamente</div>';
}
?>
```

Figura 5.3.2.5 Código PHP de alertas

La siguiente opción que encontramos en la página principal es la de añadir informe nuevo, es un botón que lleva al usuario a un formulario llamado *addInforme.php*. Se enviará por URL el *id escolar* del alumno al que pertenece el informe para poder almacenarlo en la base de datos más adelante. Este botón solo aparecerá si la persona que ha iniciado sesión es administrado o es el tutor del alumno cuyo informe se quiere realizar.

```php
<!-- si es el tutor o si es admin -->
<?php if ($found == 1 || strpos($_SESSION['rol'], "1") !== false) { ?>
    <div class="col">
        <a name="nuevo" class="btn btn-primary" href = '<?php echo $url; ?>addInforme.php?idescolar=<?php echo $_GET['idescolar']; ?>&id_login=<?php echo $_GET['id_login']; ?>'">Añadir nuevo informe</a>
    </div><br>
<?php } ?>
```

Figura 5.3.2.6 Código HTML botón añadir informe

En este formulario, el usuario tendrá que agregar los datos del nuevo informe que desea añadir y pulsar el botón *Añadir*. Si en la etiqueta del campo hay un asterisco (*) significa que el campo es obligatorio de rellenar, es decir, no se podrán guardar los datos a no ser que todos los campos con asterisco estén rellenos. Se obtiene el listado de profesores insertados en la base de datos para seleccionar posteriormente el que realiza uno de los apartados del informe.

```php
<?php
$lprof = mysqli_query($connect, "SELECT * FROM profesor") or die(mysqli_connect_error()); //obtiene todos los profesores
$p = mysqli_fetch_assoc($lprof);
?>
```

Figura 5.3.2.7 Consulta SELECT asignaturas y profesores

Para que el usuario pueda seleccionar cómodamente el profesor que imparte la asignatura, en la mayoría de los casos es él mismo, el sistema dispondrá de una lista desplegable para los profesores. Ya que es el tutor el que inicia el informe, en esta página solo se dispondrá de un apartado para que él lo rellene con la información necesaria, tanto el día de la tutoría como la fecha máxima para realizar el informe.

La asignatura se escribirá en su cuadro correspondiente y se elegirán las fechas de la tutoría y la fecha máxima para realizar el informe.

```
<form class=" row g-3" action="lib/saveAddInfTut.php?id_login=<?php echo $_GET['id_login']; ?>" method="POST" enctype="multipart/form-data">
    <input type="hidden" name="alumno" value="<?php echo $_GET['idescolar']; ?>" />
    <div class="col-md-4">
        <label for="titulo" class="form-label">Introduzca el t&iacute;tulo del informe:* </label>
        <input name="titulo" class="form-control" type="text" required/>
    </div>
    <div class="col-md-4">
        <label for="fecha_tutoria" class="form-label">Seleccione la fecha de la tutoría:* </label>
        <input name="fecha_tutoria" class="form-control" type="date" required/>
    </div>
    <div class="col-md-4">
        <label for="fecha_fin_informe" class="form-label">Seleccione la fecha máxima para realizar el informe:* </label>
        <input name="fecha_fin_informe" class="form-control" type="date" required/>
    </div>
    <div class="col-md-6">
        <label for="asignatura" class="form-label">Introduzca la asignatura:* </label>
        <input type="text" class="form-control" name="asignatura" required/>
    </div>
    <div class="col-md-6">
        <label for="profesor" class="form-label">Seleccione el profesor:* </label>
        <select name="profesor" class="form-select" required>
            <option>Elige</option>
            <?php while ($p != NULL) { ?>
                <option value="<?php echo $p['profesor'] ?>"><?php echo $p['profesor'] ?></option>
                <?php
                $p = mysqli_fetch_assoc($lprof);
            }
            ?>
        </select>
    </div>
    <div class="col-md-12">
        <label for="datos" class="form-label">Rellene el informe:* </label>
        <textarea name="datos" class="form-control" style="height: 100px" required></textarea>
    </div>
    <!-- boton submit formulario -->
    <div class="d-flex justify-content-end bd-highlight mb-3">
        <button type="submit" class="btn btn-primary">A&ntilde;adir</button>
    </div>
```

Figura 5.3.2.8Código HTML formulario añadir informe

El formulario también posee un botón llamado Volver a Gestión de informes de tutoría el cual redirecciona a la página *gInformeAlumno.php* del alumno en el que se encontraba antes de crear el formulario. Tras pulsar el botón *Añadir* el sistema llama al archivo *saveAddInfTut.php* que es el encargado de añadir a la base de datos.

Ya que se ha iniciado un informe de tutoría, el sistema debe avisar a los profesores que deben rellenar este informe mediante un correo electrónico. Para ello se ha utilizado la librería *PHPMailer*, instalada en el proyecto mediante *Composer* que es una herramienta que permite instalar librerías con sus respectivas dependencias y tenerlas actualizas en el proyecto sin tener que volver a descargarlas. Esta librería, cuyo código fuente se encuentra en el siguiente enlace de GitHub, es una librería bajo licencia LGPL 2.1 que proporciona sentencias para la creación y envío de un correo utilizando SMTP.

Para poder enviar los correos, se ha creado una cuenta de correo Gmail, que se podrá cambiar a la que el centro educativo desee modificando en el archivo *BDConnection.php* las variables *correo* y *correopwd*, que son el correo y la contraseña respectivamente. Lo primero que realiza el sistema es cargar el archivo *BDConnection.php* y el archivo *autoload.php* que es el que contiene las dependencias de la librería descrita anteriormente.

```php
require('../database/BDConnection.php');
require_once '../vendor/autoload.php';

use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;
use PHPMailer\PHPMailer\SMTP;
```

Figura 5.3.2.9 Carga de librerías

A continuación se seleccionarán los datos necesarios del usuario que ha iniciado sesión y de los profesores para enviar el email.

```php
$ql = mysqli_query($connect, "SELECT * FROM login WHERE id_login='" . $_GET['id_login'] . "'");
$l = mysqli_fetch_assoc($ql);

//obtiene la unidad del alumno al que se le va a hacer el informe
$qu = mysqli_query($connect, "SELECT unidad FROM alumno WHERE id_escolar='" . $_POST['alumno'] . "'")or die(mysqli_connect_error() . "no coge la unidad");
$u = mysqli_fetch_assoc($qu);

//seleccionar todos los profesores que dan clase en la unidad del alumno
//seleccionar los emails de los profesores que dan clase al alumno y obtiene todo en un array
$query = mysqli_query($connect, 'SELECT profesor, email FROM profesor WHERE profesor IN (SELECT profesor FROM materia WHERE unidad="' . $u['unidad'] . '")') or die(mysqli_connect_error());
$emails = mysqli_fetch_assoc($query);
```

Figura 5.3.2.10 Obtención de datos

Lo próximo que hace el sistema, es la configuración de las propiedades del correo para poder enviarlo con SMTP.

```php
//crea la clase para enviar correo con sus propiedades
$mail = new PHPMailer();
$mail->isSMTP();
$mail->SMTPDebug = SMTP::DEBUG_SERVER;
$mail->Host = 'smtp.gmail.com';
$mail->Port = 587;
$mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
$mail->SMTPAuth = true;
$mail->Username = $correo;
$mail->Password = $correopwd;
```

Figura 5.3.2.11 Creación del correo

Después, se guardará de quién proviene el correo, en este caso será del tutor del alumno, y se crea el asunto y el cuerpo del mensaje. Además se añaden al listado de envío, todos los profesores que pertenecen a la unidad en la que se encuentra el alumno. Para finalizar, se envía el email y en caso de que haya habido algún error se mostrará la información necesaria con el motivo por el que no se envió el correo.

```php
//poner el correo del que ha iniciado sesión como el que lo manda ya que debe ser su tutor el que ha iniciado el informe
$mail->setFrom($_SESSION['email'], $_SESSION['nombre']);
//redactar correo
$mail->Subject = "Nuevo informe de alumnado";
$body = "Se ha creado un nuevo informe de tutoría para el alumno con ID escolar " . $_POST['alumno']
        . ". La fecha máxima para realizar el informe es " . $_POST['fecha_fin_informe']
        . " por favor rellénelo lo antes posible. Gracias."; // Cuerpo del mensaje
$mail->Body = $body; // Fija el cuerpo del mensaje
//añadir correos de los profesores al listado para enviar
while ($emails != NULL) {
   $mail->AddAddress($emails['email'], utf8_encode($emails['profesor']));
   $emails = mysqli_fetch_assoc($query);
}
//se envía el correo
$enviado = $mail->Send();
if (!$enviado) {
    echo 'Mailer Error: ' . $mail->ErrorInfo;
}
```

Figura 5.3.2.12 Creación del mensaje del correo y envío

Para terminar, el sistema crea la consulta en la que se insertan con los datos obtenidos mediante el método POST que pertenecen a la tabla *informealumno*, esto es el título, la fecha de la tutoría y la fecha máxima para realizar el informe. Una vez se insertan estos datos, se obtiene el id de la inserción para poder relacionar el informe realizado por el tutor con estos datos.

```php
//guarda los primeros datos en la tabla principal de la bd
$query = sprintf('INSERT INTO informealumno (id_alumno, fecha_tutoria, fecha_max_informe, nombre) VALUES ("%s", "%s", "%s", "%s")',
       $_POST['alumno'], $_POST['fecha_tutoria'], $_POST['fecha_fin_informe'], $_POST['titulo']); //consulta
echo $res = mysqli_query($connect, $query) or die(mysqli_connect_error() . 'error al insertar el informe'); //ejecuta la consulta
$id = mysqli_insert_id($connect); //obtiene la id del utlimo elemento que se ha insertado
```

Figura 5.3.2.13 Consulta INSERT informealumno

Tras realizar esta consulta, el sistema obtiene todos los apartados de informes que se han insertado en *informeaux*. Si no se ha insertado ninguno, el sistema lo inserta sin necesidad de poner ningún id ya que es un campo auto incremental. Si se ha insertado alguno se coge la id más grande, es decir la del último apartado de informe que se insertó, se le suma 1 y es la id del apartado que se insertará a continuación. El sistema ejecuta la consulta y redirige a *gInformeAlumno.php* del alumno en el que se encontraba antes de realizar el informe y manda por URL el parámetro *add* para que muestre la alerta correcta.

```php
//obtiene los que hay en informeaux
$q = mysqli_query($connect, "SELECT * FROM informeaux")or die(mysqli_connect_error() . 'error en select informeaux');
if (mysqli_num_rows($q) == 0) { // si no hay ninguno insertado, se inserta normal
//guarda el informe del profesor en la tabla auxiliar con el id de los datos de la tabla original
    $qaux = sprintf('INSERT INTO informeaux (id_informe, asignatura, profesor, informe) VALUES ("%s", "%s", "%s", "%s")',
           $id, $_POST['asignatura'], $_POST['profesor'], $_POST['datos']); //consulta
} else { //si hay alguno insertado
    $qiaux = mysqli_query($connect, "SELECT MAX(id_aux) AS id_auxb FROM informeaux")or die(mysqli_connect_error());
    $iaux = mysqli_fetch_assoc($qiaux);
    $qaux = sprintf('INSERT INTO informeaux (id_aux, id_informe, asignatura, profesor, informe) VALUES ("%s","%s", "%s", "%s", "%s")',
           $iaux['id_auxb']+1, $id, $_POST['asignatura'], $_POST['profesor'], $_POST['datos']); //consulta
}
echo $r = mysqli_query($connect, $qaux) or die(mysqli_connect_error() . 'error al insertar el informe aux'); //ejecuta la consulta

$urlr = $url . "gInformeAlumno.php?add=1&idescolar=" . $_POST['alumno'] . "&id_login=" . $_GET['id_login']; //url a la que redireccionar
header('Location: ' . $urlr); //redirecciona a la url anterior
```

Figura 5.3.2.14 Consulta INSERT informeaux

A continuación se muestra el formulario tal y como lo vería el usuario:

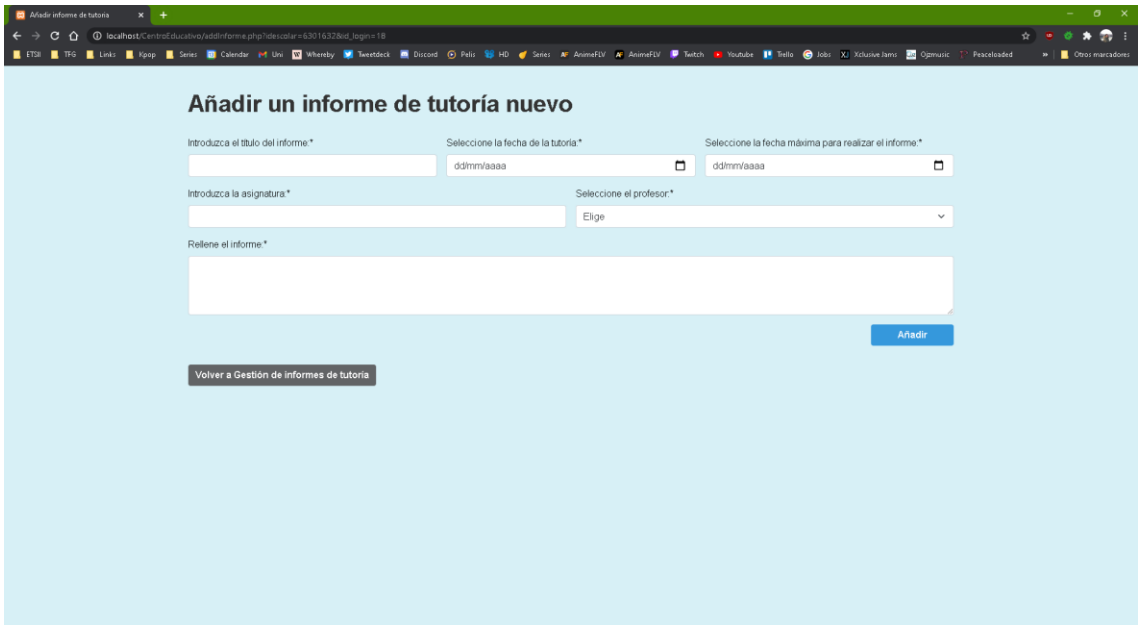

Figura 5.3.2.15 Formulario añadir informe

Tras el botón de Añadir nuevo informe, aparece la tabla de datos. Para rellenar esta tabla, el sistema realiza una consulta SELECT que obtiene la lista de todas los informes que se han introducido anteriormente. En caso de que la consulta no devuelva nada, la tabla solo mostrará la cabecera.

Una vez tenemos la lista, por cada informe se mostrará una fila con el título, el profesor y la asignatura, además de cuatro opciones que se detallarán más adelante.

```
<?php
$linf = mysqli_query($connect, "SELECT * FROM informealumno WHERE id_alumno='" . $_GET['idescolar'] . "'") or die(mysqli_connect_error());
$row = mysqli_fetch_assoc($linf);
?>
<table>
    <thead>
        <tr>
            <th>T&iacute;tulo</th>
            <th>Fecha entrega</th>
            <th>Fecha tutoria</th>
            <th></th>
            <th></th>
            <th></th>
            <th></th>
        </tr>
    </thead>
    <?php while ($row != NULL) { ?>
        <tbody>
            <tr>
                <td><?php echo $row['nombre']; ?></td>
                <td><?php echo $row['fecha_max_informe']; ?></td>
                <td><?php echo $row['fecha_tutoria']; ?></td>
```

Figura 5.3.2.16 Código HTML tabla de datos

Estas tres opciones, mencionadas anteriormente, que hacen uso de los datos son *Editar*, *Consultar, Descargar* y *Eliminar*. Cada opción es un botón que, o bien te lleva a un formulario o bien realiza la acción que su propio nombre indica. La opción de editar solo estará disponible si el usuario es admin o es profesor del alumno cuyo informe se va a editar, en caso de la opción de descargar, cuya función es crear un pdf con el informe completo del alumno, y de la opción de eliminar solo estarán disponibles para el usuario que sea admin o para el tutor del alumno.

```
<?php if ($foundp == 1 || strpos($l['rol'], "1") !== false) { ?>
    <td><a class="btn btn-primary" href='<?php echo $url; ?>editInforme.php?id=<?php echo $row['id_informe']; ?>&id_login=<?php echo $_GET['id_login']; ?>'>Editar</a></td>
<?php } else { ?>
    <td><button class="btn btn-primary" disabled>Editar</button></td>
<?php } ?>
<td><a class="btn btn-primary" href='<?php echo $url; ?>showInforme.php?id=<?php echo $row['id_informe']; ?>&id_login=<?php echo $_GET['id_login']; ?>'>Consultar</a></td>
<?php if ($found == 1 || strpos($l['rol'], "1") !== false) { ?>
    <td><a class="btn btn-primary" href='<?php echo $url; ?>lib/createInformePDF.php?id=<?php echo $row['id_informe']; ?>'>Descargar</a></td>
<?php } else { ?>
    <td><button class="btn btn-primary" disabled>Descargar</button></td>
<?php } ?>
<?php if ($found == 1 || strpos($l['rol'], "1") !== false) { ?>
    <td><button class="delete btn btn-danger" id="<?= $row['id_informe'] ?>" data-id="<?= $row['id_informe'] ?>" >Eliminar</button></td>
<?php } else { ?>
    <td><button class="btn btn-danger" disabled>Eliminar</button></td>
<?php } ?>
```

Figura 5.3.2.17 Código HTML opciones de tabla

Si el usuario pulsa el botón Editar, se muestra la página *editInforme.php* a la cual se le ha enviado la *id* de la unidad que se quiere editar a través de la URL y el sistema realizará la consulta SELECT para obtener tanto los datos del informe que hay en la base de datos.

```
<?php
$id = $_GET['id']; //coge el id del informe de la url
$query = mysqli_query($connect, 'SELECT * FROM informealumno WHERE id_informe="' . $id . '"') or die(mysqli_connect_error() . "no coge el id");
$row = mysqli_fetch_assoc($query);
$q = mysqli_query($connect, 'SELECT * FROM informeaux WHERE id_informe="' . $id . '"') or die(mysqli_connect_error() . "no coge el id"); //obtie
$aux = mysqli_fetch_assoc($q);
?>
```

Figura 5.3.2.18 Consulta SELECT de un informe

Una vez que tenemos los datos del informe, se realiza un formulario con forma de tabla la cual ya está rellena. Para el profesor y la asignatura se mostrarán en el cuadro de texto correspondiente y el informe en su textarea. En cuanto a las fechas y al título del informe, se mostrarán encima de tabla.

```
<form id="form" class=" row g-3" method="POST" action="lib/saveEditInfTut.php?id=<?=$id?>&id_login=<?=$_GET['id_login']?>" enctype="multipart/form-data">
    <input type="hidden" name="id_alumno" value="<?= $row['id_alumno']; ?>"/>
    <div class="col-md-4">
        <label for="titulo" class="form-label">Introduzca el t&iacute;tulo del informe:* </label>
        <input name="titulo" class="form-control" type="text" required value="<?= $row['nombre']; ?>"/>
    </div>
    <div class="col-md-4">
        <label for="fecha_tutoria" class="form-label">Seleccione la fecha de la tutoria:* </label>
        <input name="fecha_tutoria" class="form-control" type="date" required value="<?= $row['fecha_tutoria']; ?>"/>
    </div>
    <div class="col-md-4">
        <label for="fecha_fin_informe" class="form-label">Seleccione la fecha máxima para realizar el informe:* </label>
        <input name="fecha_fin_informe" class="form-control" type="date" required value="<?= $row['fecha_max_informe']; ?>"/>
    </div>
    <table style="text-align: center" class="tableFixHead">
        <thead>
            <tr>
                <th style="width:25%;">Asignatura</th>
                <th style="width:25%;">Profesor</th>
                <th style="width:50%;">Informe</th>
            </tr>
        </thead>
        <tbody>
            <?php
            $cont = 0; //guarda el numero de apartados informes que se pueden rellenar, como mucho serán 12
            while ($aux != NULL) {
                ?>
                <tr>
                    <td style="width:25%;"><input type="text" class="form-control" name="asignatura-<?=$aux['id_aux']?>" value="<?= $aux['asignatura']; ?>"/></td>
                    <td style="width:25%;"><input type="text" class="form-control" name="profesor-<?=$aux['id_aux']?>" value="<?= $aux['profesor']; ?>"/>
                    </td>
                    <td style="width:50%;"><textarea name="informe-<?=$aux['id_aux']?>" class="form-control" style="height: 100px" value="<?= $aux['informe']; ?>"><?= $aux['informe']; ?></textarea></td>
                </tr>
                <?php
                $cont++;
                $aux = mysqli_fetch_assoc($q);
            }
```

Figura 5.3.2.19 Código HTML formulario editar informe (1)

Además de los apartados ya rellenos, habrá varios apartados más para que el resto de profesores puedan hacer su parte del informe, en total habrá opción para rellanar hasta 12 apartados.

```
            while ($cont <= 12) {
                ?>
                <tr>
                    <td style="width:25%;"><input type="text" class="form-control" name="asignatura-<?=$cont?>"/> </td>
                    <td style="width:25%;"><input type="text" class="form-control" name="profesor-<?=$cont?>"/>
                    <td style="width:50%;"><textarea name="informe-<?=$cont?>" class="form-control" style="height: 100px"></textarea></td>
                </tr>
                <?php
                $cont++;
            }
            ?>
        </tbody>
    </table>
    <div class="d-flex justify-content-end bd-highlight mb-3">
        <button id="editar" type="submit" class="btn btn-primary" disabled>Editar</button>
    </div>
</form>
```

Figura 5.3.2.20 Código HTML formulario editar informe (2)

En el momento en el que el usuario modifique algún dato, el botón llamado *Editar* que se encuentra al final del formulario se habilitará.

Por último, el usuario pulsará el botón y el sistema llamará al archivo *saveEditInfTut.php* que realizará la consulta UPDATE con los parámetros que se obtuvieron utilizando POST y actualizará los datos del informe. Para saber qué informe tenemos que actualizar, se pasará por URL la *id* del informe.

Este archivo primero actualiza los datos de *informealumno* que son el titulo del informe, la fecha de la tutoría y la fecha máxima de realización del informe. A continuación busca todos los apartados que pertenecen a ese informe y el número de apartados que se han rellenado.

```
$id = $_GET['id']; //coge el id del informe que se edita de la url
//guarda el titulo, fecha tutoria y fecha maxima para realizar el informe en informealumno
$query = "UPDATE informealumno SET nombre='" . $_POST['titulo'] . "', fecha_tutoria='" . $_POST['fecha_tutoria'] . "', fecha_max_informe='" . $_POST['fecha_fin_informe']
    . "' WHERE id_informe='" . $id . "'"; // consulta
echo $res = mysqli_query($connect, $query) or die(mysqli_connect_error() . 'error al editar el informealumno'); //ejecuta la consulta
//obtiene todos los apartados de informeaux que pertenecen al informe principal
//y ya se habian introducido anteriormente en la base de datos
$qsia = mysqli_query($connect, "SELECT * FROM informeaux WHERE id_informe='" . $id . "'");
$nrows = mysqli_num_rows($qsia); //obtiene el numero de filas que pertenecen al informe
$sia = mysqli_fetch_assoc($qsia);
```

Figura 5.3.2.21 Consulta UPDATE de informealumno

Si hay apartados rellenos en los informes, se actualizarán con los nuevos datos de cada apartado del informe.

```
while ($sia != NULL) {
    //guarda los apartados del informe editados por los profesores en informeaux
    $quia = "UPDATE informeaux SET asignatura='" . $_POST['asignatura-' . $sia['id_aux']] . "', profesor='"
        . $_POST['profesor-' . $sia['id_aux']] . "', informe='" . $_POST['informe-' . $sia['id_aux']]
        . "' WHERE id_aux='" . $sia['id_aux'] . "'";
    echo $rquia = mysqli_query($connect, $quia) or die(mysqli_connect_error() . 'error al editar informeaux'); //ejecuta la consulta
    $sia = mysqli_fetch_assoc($qsia);
}
```

Figura 5.3.2.22 Consultas UPDATE de informeaux

Si el número de apartados que pertenecen al informe principal es menor que 12 y esos apartados se han rellenado, se insertarán en la tabla informeaux. La id de cada nuevo apartado del informe será la id del último elemento introducido, la mayor de la tabla, sumándole 1. Tras estas inserciones, el sistema redireccionará a la página principal de informes del alumnado.

```php
if ($nrows < 12) { //si hay algún apartado extra para añadir
    $cont = 12 - ($nrows); //contador para calcular el número de apartados extra que se han rellenado por primera vez
    //si las variables se ha rellenado un nuevo apartado en el formulario
    $asig = $_POST['asignatura-' . $cont];
    $prof = $_POST['profesor-' . $cont];
    $inf = $_POST['informe-' . $cont];
    if (isset($asig) && isset($prof) && isset($inf)) {
        $qmid = mysqli_query($connect, "SELECT MAX(id_aux) AS id_aux FROM informeaux");
        $mid = mysqli_fetch_assoc($qmid); //obtiene la ultima id del informe que se ha insertado
        //inserta un nuevo apartado del informe en informeaux
        $qiia = sprintf('INSERT INTO informeaux (id_aux, id_informe, asignatura, profesor, informe) VALUES ("%s", "%s", "%s", "%s", "%s")',
            $mid['id_aux'] + 1, $id, $asig, $prof, $inf);
        echo $rqiia = mysqli_query($connect, $qiia) or die(mysqli_connect_error() . 'error al insertar informeaux'); //ejecuta la consulta
    }
}

$urlr = $url . "gInformeAlumno.php?ed=1&idescolar=" . $_POST['id_alumno'] . "&id_login=" . $_GET['id_login']; //url a la que redirreciona
header('Location: ' . $urlr); //redirecciona a ginformealumno
```

Figura 5.3.2.23 Consultas INSERT de informeaux

Como el resto de formularios de esta sección, también posee un botón llamado Volver a Gestión de informes del alumno cuyo informe ha editado. A continuación se mostrará un ejemplo del formulario de edición del informe:

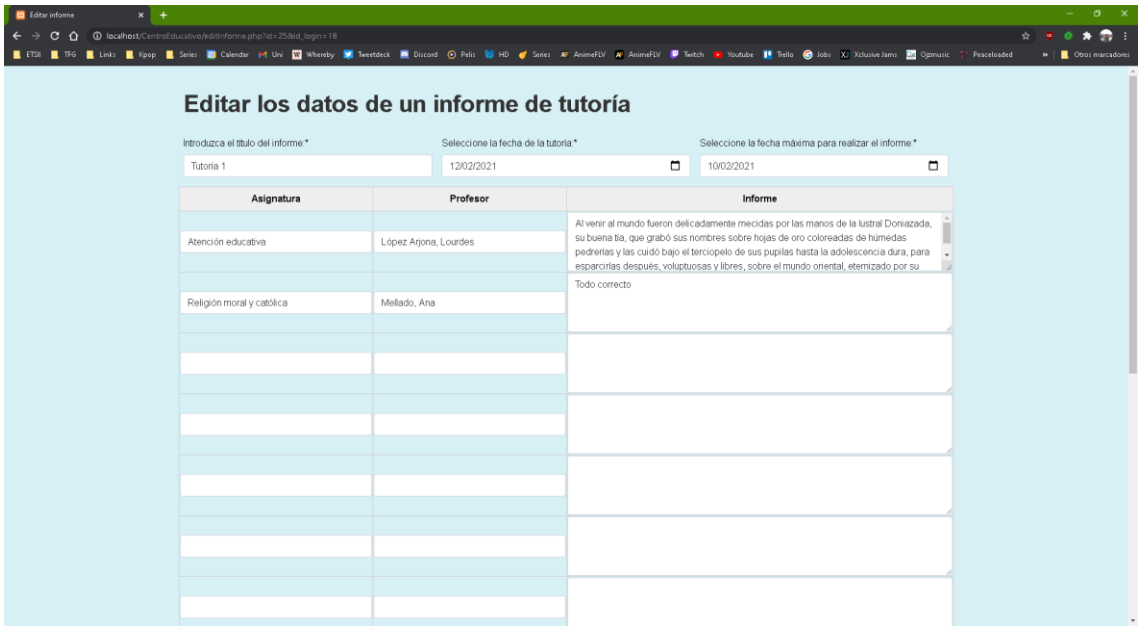

Figura 5.3.2.24 Formulario editar informe

Es la segunda opción disponible para cada fila de datos de la tabla. Cuando el usuario pulsa el botón *Consultar*, el sistema muestra la página *showInforme.php* la cual es un formulario que muestra los datos de ese informe en concreto. Para ello se envía por URL la id del informe que se desea consultar y el sistema realiza una consulta SELECT para obtener los datos de la base de datos.

Una vez se han obtenido los datos de la base de datos, el sistema muestra una tabla con los datos de cada apartado del informe. Estos datos no se pueden editar, solo se muestran a modo de consulta para el usuario.

```
<form id="form" class=" row g-3" enctype="multipart/form-data">
    <div class="col-md-4">
        <label for="titulo" class="form-label">T&iacute;tulo del informe: </label>
        <input name="titulo" class="form-control" type="text" required maxlength="9" value="<?= $row['nombre'] ?>" readonly disabled/>
    </div>
    <div class="col-md-4">
        <label for="profesor" class="form-label">Fecha de la tutoría: </label>
        <input name="profesor" class="form-control" required value="<?php echo $row['fecha_tutoria'] ?>" readonly disabled/>
    </div>
    <div class="col-md-4">
        <label for="asignatura" class="form-label">Fecha máxima realización: </label>
        <input name="asignatura" class="form-control" required readonly disabled value="<?php echo $row['fecha_max_informe'] ?>"/>
    </div>
    <div class="tableFixHead">
        <table>
            <thead>
                <tr>
                    <th style="width:25%;">Asignatura</th>
                    <th style="width:25%;">Profesor</th>
                    <th style="width:50%;">Informe</th>
                </tr>
            </thead>
            <tbody>
                <?php while ($aux != NULL) { ?>
                    <tr>
                        <td style="width:25%;"><?= $aux['profesor']; ?></td>
                        <td style="width:25%;"><?= $aux['asignatura']; ?></td>
                        <td style="width:50%;"><?= $aux['informe']; ?></td>
                    </tr>
                    <?php
                    $aux = mysqli_fetch_assoc($qi);
                }
                ?>
            </tbody>
        </table>
    </div>
</form><br>
```

Figura 5.3.2.25 Código HTML formulario consultar informe

Como el resto de formularios de esta sección, también posee un botón llamado Volver a Gestión de informes del alumno cuyo informe está consultando el usuario.

A continuación se mostrará un ejemplo del formulario de consulta de la unidad:

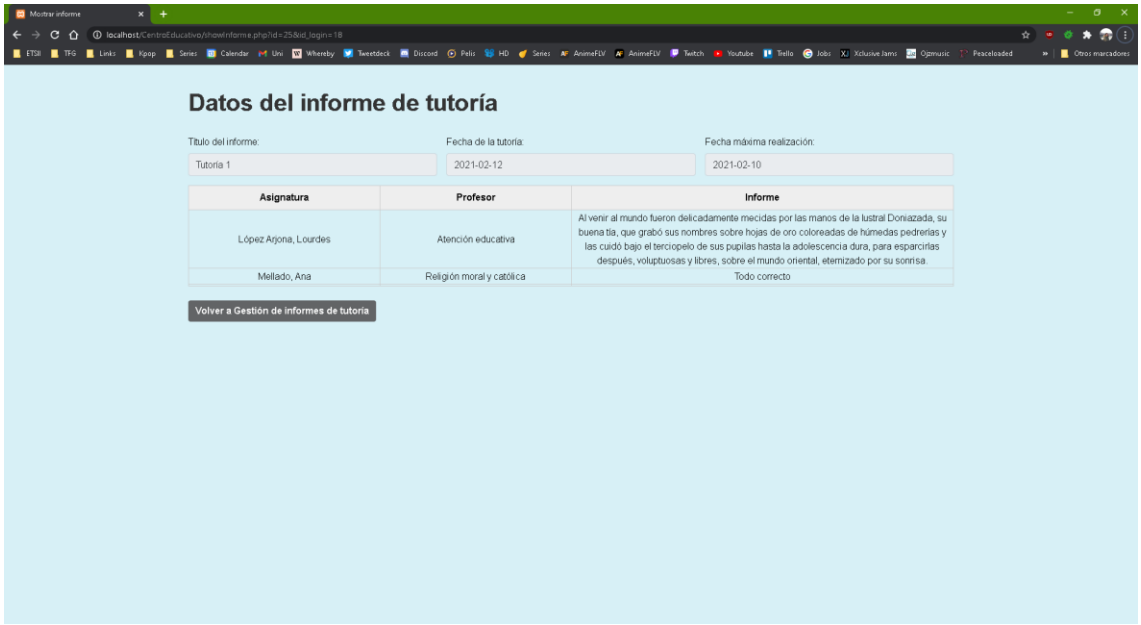

Figura 5.2.1.26 Formulario consultar informe

La siguiente opción es *Descargar*, esta opción permite crear un PDF con el informe completo del alumno y que ese PDF se descargue automáticamente. Para conseguir esto, se ha utilizado la librería HTML2PDF que se ha instalado mediante Composer, posee una licencia OSL 3.0 y su código fuente está en el siguiente enlace de GitHub.

Esta librería permite crear un PDF con solo pasarle por parámetro una página HTML o PHP. En este caso primero se enviará el archivo *informe.php* que se guardará en la variable *content*, después se creará el PDF que será vertical, tamaño A4 y en español. Para finalizar se escribirá el contenido del informe guardado en la variable *content* en el PDF y se descargará. Si hubo algún error se mostrarán los mensajes pertinentes informando de él.

```php
//cargar libreria insertada en el proyecto mediante composer
require_once '../vendor/autoload.php';

use Spipu\Html2Pdf\Html2Pdf;

try {
    ob_start();
    include dirname(__FILE__) . '/../informe.php';
    $content = ob_get_clean();

    //inicializamos el pdf mediante la clase HTML2PDF
    //le indicamos que será vertical (P), en tamño A4 y en español (es).
    $pdf = new Html2Pdf('P', 'A4', 'es', true, 'UTF-8');
    $pdf->writeHTML($content);
    $pdf->Output('informe.pdf', 'D');
} catch (Html2PdfException $e) {
    $html2pdf->clean();

    $formatter = new ExceptionFormatter($e);
    echo $formatter->getHtmlMessage();
}
```

Figura 5.2.1.27 Creación del PDF

La última acción a realizar sobre cada informe es la de eliminar. El usuario pulsará el botón *Eliminar* y el sistema ejecutará el siguiente código JavaScript que lo que hace es mostrar una mensaje preguntando si el usuario realmente desea eliminar el registro. En caso de que el usuario cierre el mensaje o haga click en la opción *Cancel*, se cerrará el mensaje y el sistema no eliminará nada.

```
// funcion para borrar filas de la tabla
$(document).ready(function () {
    $('.delete').click(function () {
        var el = this;
        //coge la id del elemento a borrar
        var deleteid = $(this).data('id');
        // muestra caja de confirmación
        bootbox.confirm("¿Realmente quiere eliminar?", function (result) {
            if (result) {
                //borra el elemento y actualiza la tabla con ajax
                $.ajax({
                    url: 'lib/deleteInforme.php',
                    type: 'POST',
                    data: {id: deleteid},
```

Figura 5.2.1.28 Código JavaScript eliminar informe (1)

En caso de que el usuario haga click en la opción *Ok*, el sistema ejecutará el archivo *deleteInforme.php*. En este archivo, el sistema recoje la id de la unidad que se quiere eliminar, se ejecuta la consulta DELETE y muestra el número 1 en caso de que se haya podido eliminar. Si no se pudo eliminar, el sistema mostará el número 0.

```
if (isset($_POST['id'])) { //si el id existe y no es null
    $id_informe = $_POST['id']; //obtiene el id del formulario
    $query = "DELETE FROM informealumno WHERE id_informe='" . $id_informe . "'"; //consulta
    $res = mysqli_query($connect, $query) or die(mysqli_connect_error() . 'error al eliminar el informe');
    echo 1;
    exit;
} else {
    echo 0;
    exit;
}
```

Figura 5.2.1.29 Código PHP eliminar informe

Si el sistema recibe el número 1 como respuesta, la fila que se ha eliminado de la base de datos se mostrará en rojo y desaparecerá de la tabla usando AJAX. En caso de que la respuesta haya sido 0, se mostrará un mensaje diciendo que no se pudo eliminar el registro.

```
success: function (response) {
    //borra el elemento
    if (response == 1) {
        $(el).closest('tr').css('background', 'tomato');
        $(el).closest('tr').fadeOut(800, function () {
            $(this).remove();
        });
    } else { // si no se ha podido eleminar muestra el mensaje
        bootbox.alert('No se ha eliminado.');
    }
}
```

Figura 5.2.1.30 Código JavaScript eliminar unidad (2)

Para terminar este apartado, justo debajo de la tabla hay un botón para volver a la página de inicio, en el caso de que el usuario quiera acceder a otro apartado de gestión.

5.3.3 Promoción automática del alumnado

Esta sección de código se ejecuta nada más iniciar sesión y entrar en la página principal (apartado 5.1.3). Lo primero que hace es cargar el archivo *BDConnection.php* y a continuación ejecuta una consulta SELECT para obtener la fecha de la última vez que se promocionó a los alumnos de curso. Además se obtiene el año en el que se promocionó por última vez a los alumnos y en el que se encuentra el usuario que ha iniciado sesión.

```php
<?php
require('database/BDConnection.php');
?>
<!DOCTYPE html>
<!-- aumentar los alumnos de curso -->
<?php
$res = mysqli_query($connect, "SELECT * FROM aumentarcurso");
$ac = mysqli_fetch_assoc($res); //obtiene la ultima vez que se hizo un aumento de curso
$hoy = explode("-", date("Y-m-d")); //año de la fecha actual
$fa = explode("-", $ac['fecha']); //año de la ultima vez que se promocionó
```

Figura 5.3.3.1 Consulta SELECT y obtención de fechas

En caso de que el año actual coincida con el año del inicio del curso escolar, el sistema no hace nada ya que se han promocionado a los alumnos anteriormente en este curso escolar. En caso de que el año de la fecha actual sea mayor o coincida con el año de la última fecha guardada en la base de datos más 1 y, además de los años, el mes de la fecha actual sea mayor que el mes de la última fecha guardada en la base de datos, el sistema ejecutará una consulta SELECT para obtener a todos los alumnos almacenados en la base de datos y promocionarlos un curso. También guarda la fecha actual como la última fecha en la que se promocionó a los alumnos.

```php
//si el año actual es igual del que se aumento el curso por ultima vez +1
//y el mes actual es mayor o igual que el mes en el que se aumento curso por ultima vez
if (strcmp($hoy[0], $fa[0]+1) >= 0 || (strcmp($hoy[0], $fa[0]+1) == 0 && strcmp($hoy[1], $fa[1]) >= 0)) {
    $aum = $ac['aumentado'] + 1;
    //guarda la fecha actual como la ultima vez que se subio de curso
    $query = mysqli_query($connect, "UPDATE aumentarcurso SET aumentado='" . $aum
            . "', fecha='" . date("Y-m-d") . "' WHERE id_ac='" . $ac['id_ac'] . "'");

    //selecciona todos los datos de la tabla alumno
    $lal = mysqli_query($connect, "SELECT * FROM alumno") or die(mysqli_connect_error());
    $row = mysqli_fetch_assoc($lal); //ejecuta la consulta
```

Figura 5.3.3.2 Consulta SELECT de alumnos y actualización de fecha

A continuación, se crean las variables *unidad* y *curso* cuya función es almacenar la unidad y el curso nuevo, en función del curso actual del alumno. Además, se realiza un bucle en el que mientras haya alumnos en el listado obtenido de la base de datos, se compare la unidad actual en el que está el alumno con un String de unidad. Si ambas unidades coinciden, se cambia las variables *unidad* y *curso* a un curso y unidad posterior, si no coinciden se pasa al siguiente *if*.

```php
$curso = "";
$unidad = "";
while ($row != NULL) {
    //si el alumno es de tres años
    if (strcmp($row['unidad'], "I3A") == 0) {
        $curso = "Cuatro años";
        $unidad = "I4A";
    }

    if (strcmp($row['unidad'], "I3B") == 0) {
        $curso = "Cuatro años";
        $unidad = "I4B";
    }
    //si el alumno es de cuatro años
    if (strcmp($row['unidad'], "I4A") == 0) {
        $curso = "Cinco años";
        $unidad = "I5A";
    }

    if (strcmp($row['unidad'], "I4B") == 0) {
        $curso = "Cinco años";
        $unidad = "I5B";
    }
```

Figura 5.3.3.3 Variables, bucle y comparación con tres y cuatro años

```php
//si el alumno es de cinco años
if (strcmp($row['unidad'], "I5A") == 0) {
    $curso = "Primero primaria";
    $unidad = "1°A";
}

if (strcmp($row['unidad'], "I5B") == 0) {
    $curso = "Primero primaria";
    $unidad = "1°B";
}
//si el alumno es de primero de primaria
if (strcmp($row['unidad'], "1°A") == 0) {
    $curso = "Segundo primaria";
    $unidad = "2°A";
}

if (strcmp($row['unidad'], "1°B") == 0) {
    $curso = "Segundo primaria";
    $unidad = "2°B";
}
//si el alumno es de segundo de primaria
if (strcmp($row['unidad'], "2°A") == 0) {
    $curso = "Tercero primaria";
    $unidad = "3°A";
}

if (strcmp($row['unidad'], "2°B") == 0) {
    $curso = "Tercero primaria";
    $unidad = "3°B";
}
```

```php
//si el alumno es de tercero de primaria
if (strcmp($row['unidad'], "3°A") == 0) {
    $curso = "Cuarto primaria";
    $unidad = "4°A";
}

if (strcmp($row['unidad'], "3°B") == 0) {
    $curso = "Cuarto primaria";
    $unidad = "4°B";
}
//si el alumno es de cuarto de primaria
if (strcmp($row['unidad'], "4°A") == 0) {
    $curso = "Quinto primaria";
    $unidad = "5°A";
}

if (strcmp($row['unidad'], "4°B") == 0) {
    $curso = "Quinto primaria";
    $unidad = "5°B";
}
//si el alumno es de quinto de primaria
if (strcmp($row['unidad'], "5°A") == 0) {
    $curso = "Sexto primaria";
    $unidad = "6°A";
}

if (strcmp($row['unidad'], "5°B") == 0) {
    $curso = "Sexto primaria";
    $unidad = "6°B";
}
```

Figura 5.3.3.4 Comparaciones con el resto de unidades

Para finalizar, en caso de que el alumno sea de sexto de primaria se borrará de la base de datos y si es de un curso distinto de sexto, se actualizará su curso y unidad utilizando dichas variables creadas y modificadas anteriormente.

```
    //si el alumno es de sexto de primaria lo elimina de la base de datos
    if (strcmp($row['unidad'], "6°A") == 0 || strcmp($row['unidad'], "6°B") == 0) {
        $query = "DELETE FROM alumno WHERE id_escolar='" . $row['id_escolar'] . "'";
        echo $res = mysqli_query($connect, $query) or die(mysqli_connect_error()
            . 'error al eliminar el alumno');
    } else { //si es de otro curso, lo promociona
        $q = mysqli_query($connect, "UPDATE alumno SET curso='" . $curso . "', unidad='"
            . $unidad . "' WHERE id_escolar='" . $row['id_escolar'] . "'");
    }
    $row = mysqli_fetch_assoc($lal); //coge otra linea de las buscadas anteriormente
}
```

Figura 5.3.3.5 Comparación con sexto y borrado o actualización de la base de datos

5.3.4 Diseño de la web con CSS y Bootstrap

A continuación, se describirá el diseño de las páginas web para ello se ha resumido en cuatro apartados. El apartado 5.3.4.3 engloba todas las páginas principales de cada sección de Gestión de las que dispone la herramienta, es decir, que se describirá una página y el resto serán exactamente iguales.

El apartado 5.3.4.4 engloba todos los formularios de añadir, editar y mostrar datos de todos las secciones de Gestión de las que dispone la herramienta y al igual que en el apartado anterior, también se describirá un ejemplo de cada tipo de formulario y el resto serán todos iguales.

En cada página, se introducen las siguientes líneas en la cabecera, y en el cuerpo web para poder utilizar tanto las librerías de CSS como de Bootstrap.

```
<link rel="stylesheet" href="css/bootstrap.min.css" crossorigin="anonymous">
<script src="js/bootstrap.min.js"></script>
```

Figura 5.3.4.1 Líneas cabecera página web

```
<script src="js/popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
<script src="js/bootstrap.min.js" integrity="sha384-oesi62hOLfzrys4LxRF63OJCXdXDipiYWBnvTl9Y9/TRlw5xlKIEHpNyvvDShgf/" crossorigin="anonymous"></script>
```

Figura 5.3.4.2 Líneas cuerpo página web

Para eliminar datos de la tabla y que salga el mensaje que pide al usuario si quiere eliminar el dato, es necesario añadir las siguientes líneas en la cabecera de la web:

```
<script src="js/bootbox.min.js"></script>
<script src="https://unpkg.com/@popperjs/core@2/dist/umd/popper.js"></script>
```

Figura 5.3.4.3 Líneas cabecera para eliminar datos

Excepto la página principal, el resto de páginas tiene la letra negra, cuya fuente es Roboto, sans-serif y tiene un color de fondo cuyo código hexadecimal es d7f0f6. Además el título de la página es en negrita y de color negro (código hexadecimal 333).

```css
body {
    color: #000;
    background: #d7f0f6;
    font-family: 'Roboto', sans-serif;
}
```

```css
.container h1  {
    color: #333;
    font-weight: bold;
    margin-top: 0;
}
```

Figura 5.3.4.4 Diseño general formularios

5.3.4.1 Formularios de registro e inicio de sesión

Ambos formularios tienen el mismo diseño y estilo. Por lo que solo vamos a detallar uno de ellos, el formulario de inicio de sesión. En este caso centra el formulario en el ancho de la página y crea un cuadro blanco alrededor del formulario al cual le pone sombreado debajo. En cuanto a los botones, tiene el color predeterminado de bootstrap al botón primario y además cuando se coloca el cursor encima, este se oscurece un poco. A continuación se muestra el código CSS:

```css
.signup-form {
    width: 400px;
    margin: 30px auto;
    align-items: center;
    justify-content: center;
    min-height: 100vh;
    display: block;
}
.signup-form form {
    color: #999;
    border-radius: 3px;
    margin-bottom: 15px;
    background: #fff;
    box-shadow: 0px 2px 2px rgba(0, 0, 0, 0.3);
    padding: 30px;
}
.signup-form h2  {
    color: #333;
    font-weight: bold;
    margin-top: 0;
}
```

```css
.signup-form .btn {
    font-size: 16px;
    font-weight: bold;
    background: #3598dc;
    border: none;
    min-width: 140px;
}
.signup-form .btn:hover, .signup-form .btn:focus {
    background: #2389cd !important;
    outline: none;
}
.signup-form .hint-text  {
    padding-bottom: 15px;
    text-align: center;
}
```

Figura 5.3.4.5 Código CSS formularios registro e inicio de sesión

5.3.4.2 Página principal

Esta es la única página que no tiene el mismo fondo que el resto, en este caso se ha optado por una imagen de colegio sobre la cual se han añadido los botones a las páginas principales de gestión, el botón del manual de usuario y el encabezado que da nombre a la página.

```css
body {
    background-image: url("img/anime-classroom-background.jpg");
    background-size: cover;
    background-repeat: no-repeat;
}
```

Figura 5.3.4.6 Código CSS fondo de la página principal

Para el encabezado, las letras son blancas y en mayúsculas con el fin de resaltarlas más, sobre un fondo azul semitransparentes. El código CSS del encabezado con el fondo y el título es el siguiente:

```css
h1  {
    font-family: 'Roboto', sans-serif;
    color: #fff;
    font-weight: bold;
    margin-top: 10px;
    font-size: 80px;
    text-transform: uppercase;
}
```

```css
.neg{
    background-color: #386FB1;
    width: 100vw;
    height: 25vh;
    display: flex;
    justify-content: center;
    align-items: center;
    opacity: 0.80;
}
```

Figura 5.3.4.7 Código CSS encabezado

En cuanto a los botones principales serán de color azul, semitransparentes y con letra en blanco, además si se pone el cursor encima el color cambiará a verde para que le quede claro al usuario sobre qué botón está. El código CSS es el siguiente:

```css
.btn {
    font-size: 16px;
    font-weight: bold;
    background: #386FB1;
    min-width: 140px;
    border: none;
    border-radius: 20px;
    justify-content: center;
    align-items: center;
}
```

```css
.btn:hover, .signup-form .btn:focus {
    background: #374b3c !important;
    outline: none;
}
.container .btn {
    text-align: center;
    float:none;
    font-size: 36px;
    width: 36em;
    height: 2.25em;
    opacity: .75;
}
```

Figura 5.3.4.8 Código CSS botones principales

En cuanto al botón de cerrar sesión, se encuentra en la esquina superior derecha también es de color azul y con letras blancas y al igual que el resto, al poner el cursor encima cambia de color.

```css
.container-c .btn {
    line-height: 12px;
    margin-top: 40px;
    margin-right: 35px;
    position:absolute;
    right: 0;
    top: 0;
}
```

```css
.btn-secondary{
    font-size: 20px;
    font-weight: bold;
    background: #3E82FB;
    border: none;
    min-width: 140px;
    width: 8em;
    height: 2em;
}
```

Figura 5.3.4.9 Código CSS botón cerrar sesión

5.3.4.3 Páginas principales de Gestión

El fondo de estas páginas ya se detalló al principio de este apartado, por lo que esta sección se centrará en la tabla que contiene los datos y los botones. En cuanto al buscador y al importador de archivos, su diseño lo realiza bootstrap, solo se utilizó la clase que viene especificada en el código HTML antes descrito. Las etiquetas tienen fuente Helvetica, 20px en negrita y su código CSS es el siguiente:

```
.container label{
    width: 400px;
    font-weight:bold;
    font-size:20px;
    font-color:#ffffff;
    font-family:'Helvetica','Verdana','Monaco',sans-serif;
}
```

Figura 5.3.4.10 Código CSS etiquetas

En cuanto a la tabla principal, la cabecera se mantendrá estática y en caso de que los datos superen el tamaño dispuesto para la tabla, aparecerán scrolls para que el usuario pueda desplazar la tabla si es necesario. La cabecera tendrá fondo gris y el resto de celdas de la tabla tendrá fondo blanco.

```
.tableFixHead {
    overflow-y: auto;
    height: 500px;
}
.tableFixHead thead th {
    position: sticky;
    top: 0;
}
table {
    border-collapse: collapse;
    width: 100%;
}
```

```
th,
td {
    padding: 8px 16px;
    border: 1px solid #ccc;
    text-align: center;
}
th {
    background: #eee;
}
```

Figura 5.3.4.11 Código CSS tablas de datos

En cuanto a los botones, se basan en las clases de bootstrap, pero con CSS se ha cambiado el color de la letra para hacerla más llamativa, el tamaño de los botones y que se oscurezca el botón cada vez que el usuario pone el cursor encima.

```css
.signup-form .btn {
    font-size: 16px;
    font-weight: bold;
    background: #3598dc;
    border: none;
    min-width: 140px;
}
.signup-form .btn:hover, .signup-form .btn:focus {
    background: #0d466e !important;
    outline: none;
}
.signup-form .btn-danger{
    font-size: 16px;
    font-weight: bold;
    background: #d12d1b;
    border: none;
    min-width: 140px;
}
```

```css
.signup-form .btn-danger:hover, .signup-form .btn-danger:focus {
    background: #821208 !important;
    outline: none;
}
.signup-form .btn-secondary{
    font-size: 16px;
    font-weight: bold;
    background: #626466;
    border: none;
    min-width: 140px;
}
.signup-form .btn-secondary:hover, .signup-form .btn-secondary:focus {
    background: #423f3e !important;
    outline: none;
}
```

Figura 5.3.4.12 Código CSS botones de la página principal

5.3.4.4 Formularios de añadir, editar y mostrar

Tanto el fondo como la cabecera de estos formularios tienen el mismo código descrito en la introducción de este apartado. Además, el diseño del formulario de añadir, es el mismo que el de editar y mostrar por lo que solo se hará la descripción del diseño de uno de ellos.

En estas páginas solo se ha utilizado CSS para cambiar el diseño de los botones, el resto de campos se han diseñado utilizando las clases que bootstrap proporciona. Con respecto a los botones, se ha aumentado el tamaño, se ha puesto la letra en blanco para realzarla y, al igual que en el resto de páginas, cuando el usuario pase el cursor por encima el color de botón se pondrá más oscuro.

```css
.signup-form .btn {
    font-size: 16px;
    font-weight: bold;
    background: #3598dc;
    border: none;
    min-width: 140px;
}
.signup-form .btn:hover, .signup-form .btn:focus {
    background: #0d466e !important;
    outline: none;
}
.signup-form .btn-danger{
    font-size: 16px;
    font-weight: bold;
    background: #d12d1b;
    border: none;
    min-width: 140px;
}
```

```css
.signup-form .btn-danger:hover, .signup-form .btn-danger:focus {
    background: #821208 !important;
    outline: none;
}
.signup-form .btn-secondary{
    font-size: 16px;
    font-weight: bold;
    background: #626466;
    border: none;
    min-width: 140px;
}
.signup-form .btn-secondary:hover, .signup-form .btn-secondary:focus {
    background: #423f3e !important;
    outline: none;
}
```

Figura 5.3.4.13 Código CSS botones de formularios

6

Conclusiones

Para finalizar esta memoria se procederá a analizar, una vez terminados el desarrollo y diseño de este proyecto, los objetivos y conclusiones finales que se pueden obtener tras el proceso. Además, se especificarán las posibles líneas futuras mediante las cuales se puede expandir esta aplicación web.

6.1 Objetivos cumplidos y conclusiones

Para empezar se analizarán los objetivos cumplidos desde el punto de vista del proyecto como una aplicación web.

El objetivo de esta aplicación siempre fue crear un sistema gestor que permitiera a los docentes de un centro educativo, importar los datos del alumno, profesores y asignaturas, gestionar las ausencias de los profesores e informatizar los informes de tutoría.

Tras varias reuniones con el cliente, se obtuvo una gran cantidad de requisitos aunque no todos eran necesarios para cumplir el objetivo principal de la aplicación ni poseían la misma importancia, por lo que se decidió darles prioridad y organizarlos en función de ésta. Tras esto, se comprobó que no era posible cumplir todos los requisitos en este proyecto dada la limitación temporal que tiene el TFG.

Por ello, se seleccionó un conjunto de requisitos para realizar sus respectivos casos de uso, los cuales se encuentran en el apartado 4.2.1, y sobre los que se ha desarrollado gran parte de la web. Dado que esta memoria debe ser la base para futuras ampliaciones del proyecto, se ha decidido incluir casos de uso de funcionalidades que no han sido implementadas en este TFG.

Además, todos los requisitos cuya prioridad era alta y gran parte de los requisitos cuya prioridad era moderada han sido implementados de manera satisfactoria en la aplicación web, utilizando PHP para el acceso al servidor y base de datos y HTML junto con JavaScript para la parte visible de la aplicación web.

En cuanto al punto de vista académico, este trabajo fin de grado es fruto de una elaboración personal, en la que se han aplicado conocimientos técnicos y teóricos aprendidos en la titulación que se ha cursado para analizar el problema y diseñar y desarrollar una solución acorde a los requisitos detallados por el cliente.

Dicho esto y con los resultados obtenidos, se afirma que se han cumplido los objetivos académicos del trabajo fin de grado y que la aplicación web se ha desarrollado con éxito, cumpliendo el objetivo principal, atendiendo a gran parte los requisitos impuestos por el cliente y dentro del plazo precisado.

En cuanto a la evaluación de esta herramienta, dada la situación actual de la limitación de la movilidad y la dificultad en cuanto al acceso a centros educativos a causa del COVID-19, no ha sido posible instalar esta aplicación en el centro educativo y que el cliente la probara y diera realimentación útil que nos hubiera permitido evaluar mejor la herramienta desarrollada. Aunque este era un objetivo importante de este TFG no ha podido ser cumplido. Para paliar de alguna manera esta limitación se han desarrollado un manual de instalación y de uso detallado para que este paso pueda llevarse a cabo más adelante. También se ha preparado una máquina virtual donde se ha instalado todo el software necesario, la base de datos y el servidor web de forma que en cuanto sea posible el colegio pueda probar a fondo la aplicación de forma más fácil y rápida. Aun así, ciertas opciones de la aplicación, como la importación de datos de Séneca, han podido ser probadas gracias a ficheros que el propio centro educativo envió (sin datos válidos para asegurar la protección de datos) con el objetivo de realizar una aplicación adaptada y funcional que respetase el formato del contenido de esos archivos.

6.2 Líneas futuras

Como se ha comentado anteriormente, dada la gran cantidad de requisitos que el cliente detalló en las reuniones y que no todos se han podido implementar, este proyecto podría continuarse añadiendo los requisitos restantes para hacer más completa esta aplicación. Dichos requisitos se enumeraron al final de la sección 3.3 y se dio una idea general sobre cómo podían implementarse cada uno de ellos.

Además, sería interesante añadir una sección para el usuario en la que se diera la posibilidad de cambiar la contraseña, agregar una imagen de perfil o incluso subir en esa sección su horario personal. También podría añadirse la posibilidad de enviar un correo con una clave para que el usuario la introdujera a la hora de registrarse, a modo de verificación del correo y hacer más segura la autenticación.

Referencias

PHP

- https://openwebinars.net/blog/que-es-php/
- https://www.php.net/

JavaScript

- https://devcode.la/blog/que-es-javascript/
- https://www.w3schools.com/js/
- http://bootboxjs.com/

HTML

- https://www.w3schools.com/html/

MYSQL

- https://dev.mysql.com/doc/refman/8.0/en/

Bootstrap y CSS

- https://www.w3schools.com/css/
- https://getbootstrap.com/docs/5.0/getting-started/introduction/

Librerías PHP utilizadas

- https://github.com/PHPMailer/PHPMailer
- https://github.com/spipu/html2pdf

Apéndice A

Guía de instalación

En este apéndice se describirán los pasos para instalar la herramienta en un servidor web. Habrá varios apartados en función de si se quiere utilizar la máquina virtual proporcionada en la que ya está la herramienta instalada o si se quiere crear un servidor local en el ordenador para utilizarla.

A.1 Máquina virtual

Para instalar el programa que ejecutará la máquina virtual, es necesario descargárselo de la página https://www.virtualbox.org/wiki/Downloads. Si el usuario tiene Windows como sistema operativo, deberá pulsar el enlace marcado con 1, si tiene Linux como sistema operativo, deberá pulsar el enlace marcado con 2.

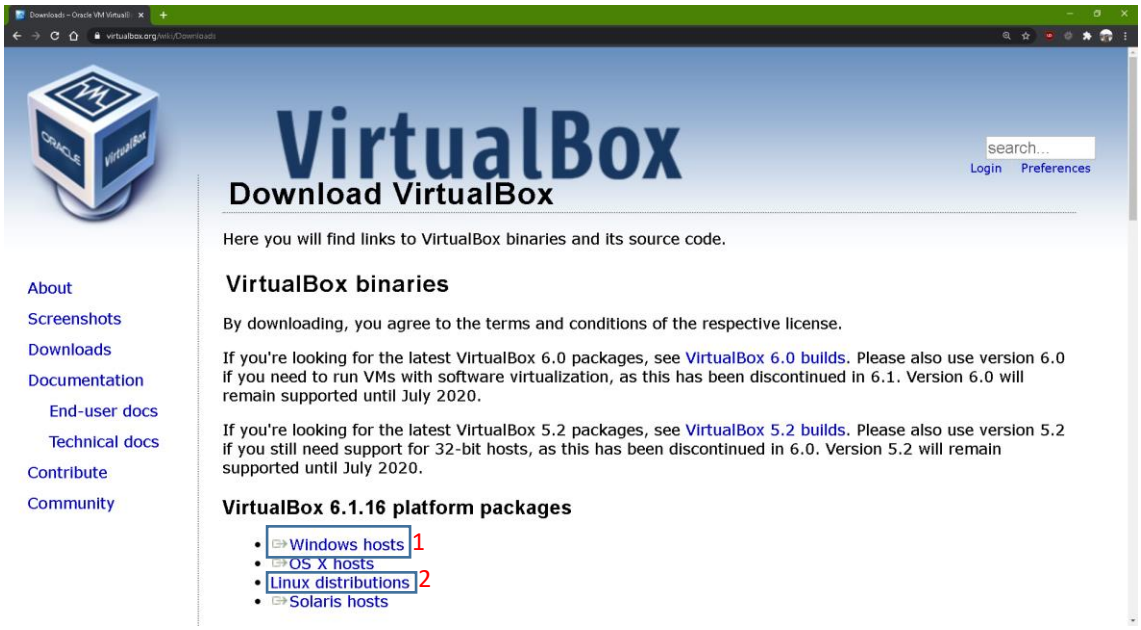

Figura A.1.1 Enlaces de descarga

En caso de que el usuario tenga Linux como sistema operativo, es decir, que haya seleccionado la opción número 2, deberá elegir la versión de Linux que tiene y pulsar en su enlace para descargarla.

Figura A.1.2 Enlaces de descarga de linux

Una vez que se ha descargado el programa se procede a la instalación del mismo, para la cual el usuario debe hacer doble click en el archivo que se acaba de descargar y debe seguir los siguientes pasos.

1. En caso de que le haya salido esta advertencia, debe hacer click en *Ejecutar.*

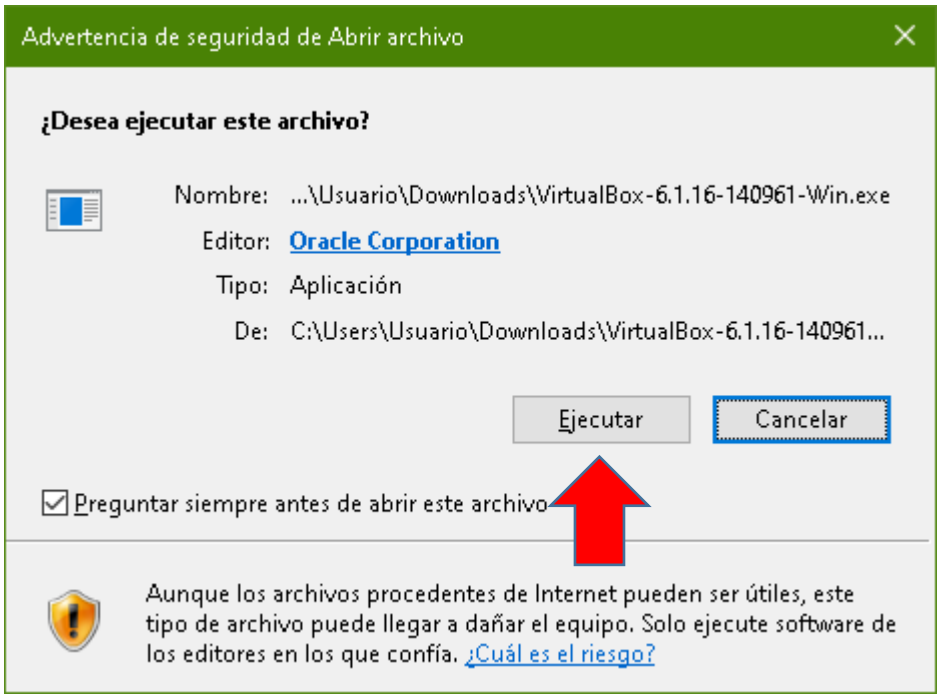

Figura A.1.3 Advertencia ejecutable

2. Una vez que aparece el instalador, debe hacerle click en *Next* hasta que aparezca la siguiente pantalla que debe hacer click en *Yes*.

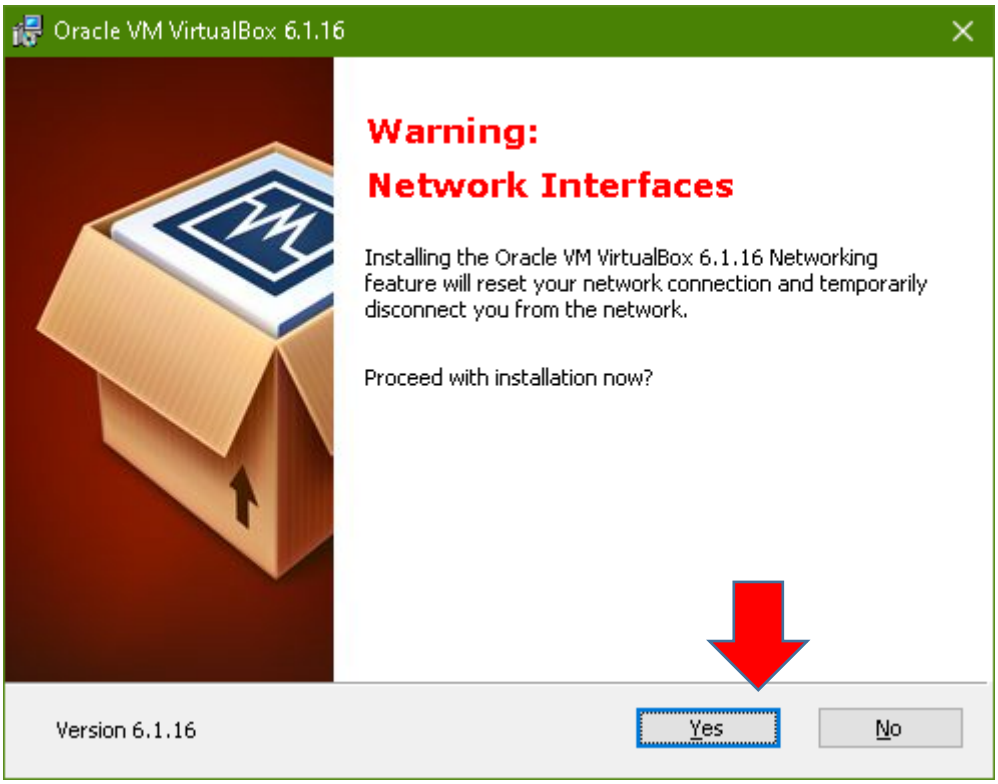

Figura A.1.4 Advertencia redes

3. En la próxima pantalla, el usuario debe hacer click en *Install*.

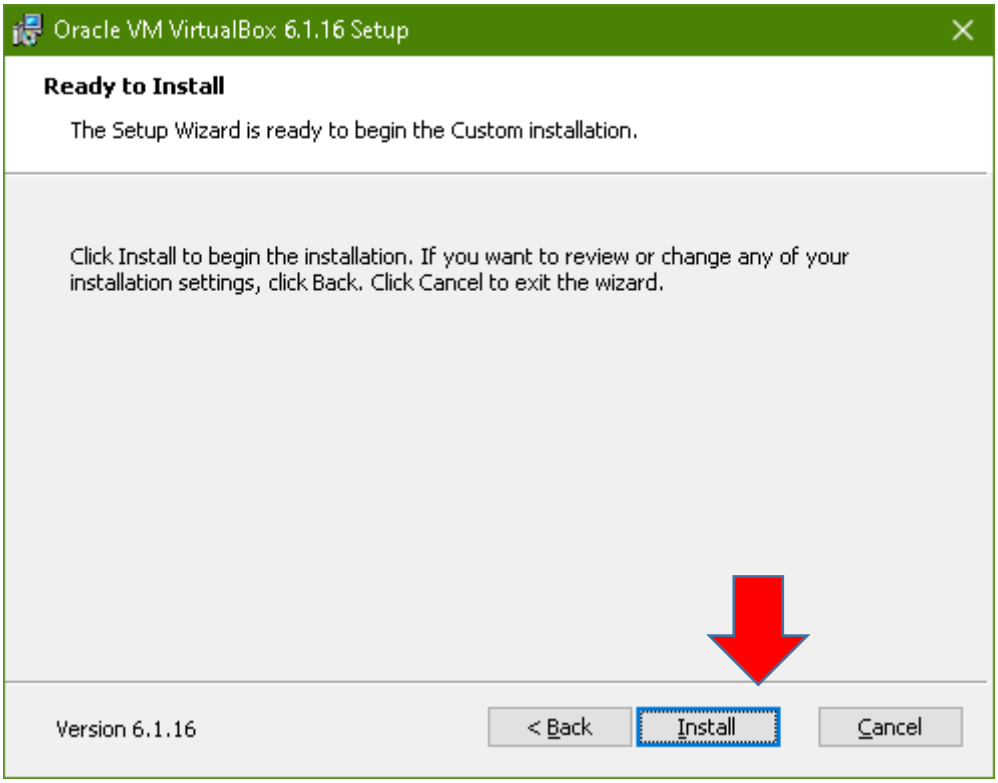

Figura A.1.5 Inicio instalación

4. El usuario debe esperar unos segundos hasta que aparezca la advertencia de Windows, debe pulsar *Sí* y esperar a que se instale el programa. Una vez instalado debe pulsar *Finish* y el programa se ejecutará.

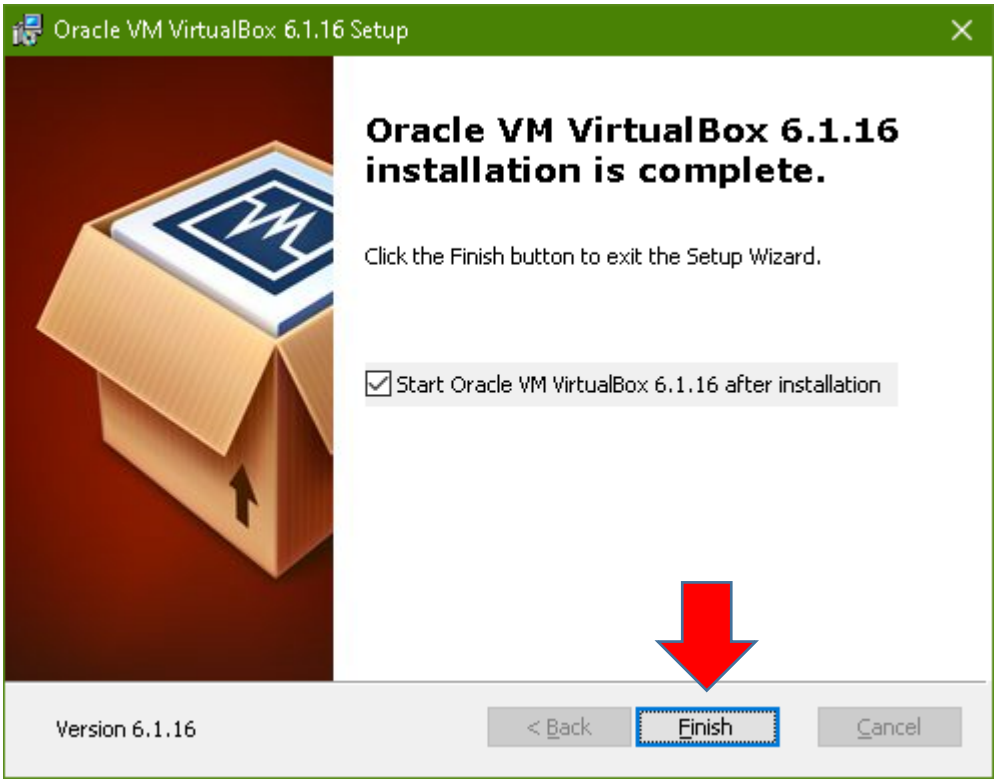

Figura A.1.6 Finalización instalación

5. En cuanto en el programa se abra, el usuario debe pulsar en *Archivo* y a continuación elegir la opción *Importar servicio virtualizado.*

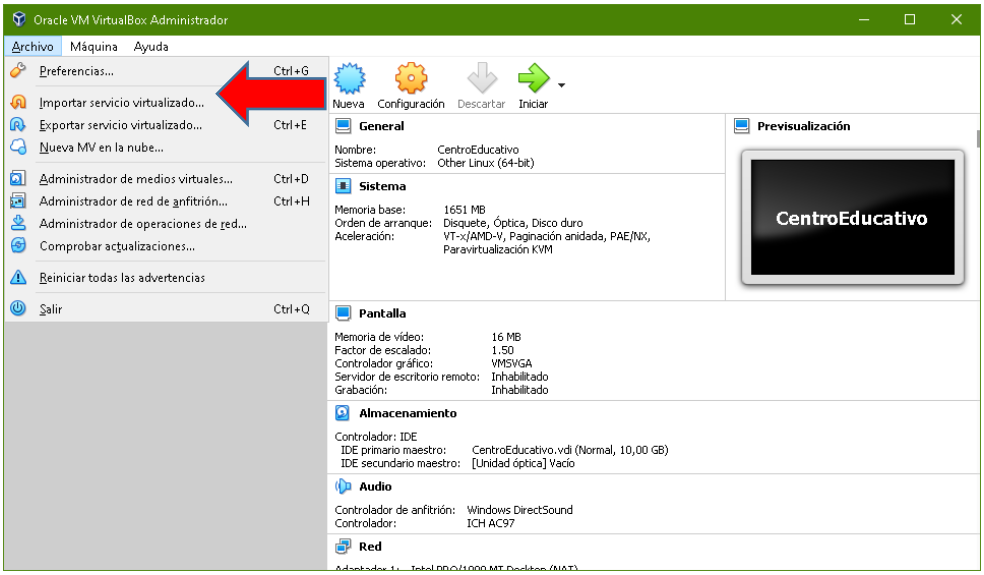

Figura A.1.7 Importar máquina virtual (1)

6. A continuación se abrirá la siguiente ventana para escoger la máquina virtual, para ello el usuario debe pulsar el botón señalado con la flecha. Tras esto se abrirá el explorador de archivos de su ordenador, ir a la ubicación en la que se encuentra la máquina virtual y pulsar el botón *Abrir.* Una vez aparece la ruta en la que se encuentra la máquina, el usuario debe pulsar el botón *Next*.

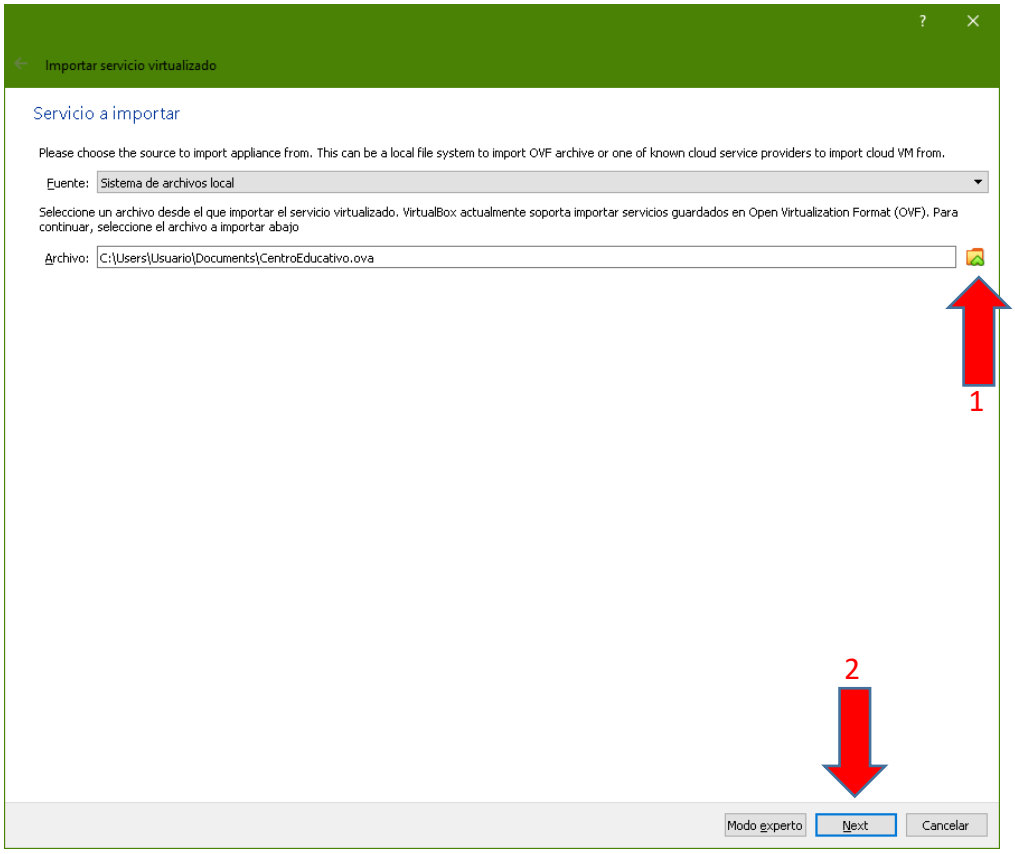

Figura A.1.8 Importar máquina virtual (2)

7. En la siguiente pantalla que se muestra, el usuario debe pulsar *importar* y esperar que la siguiente barra de progreso llegue al 100%.

Figura A.1.9 Importar máquina virtual (3)

8. Una vez se ha importado, el usuario debe tener marcada la máquina virtual (que tenga el fondo azul) y darle al botón *iniciar* y esperar que cargue.Tanto el usuario como contraseña de la máquina virtual es *centroeducativo*.

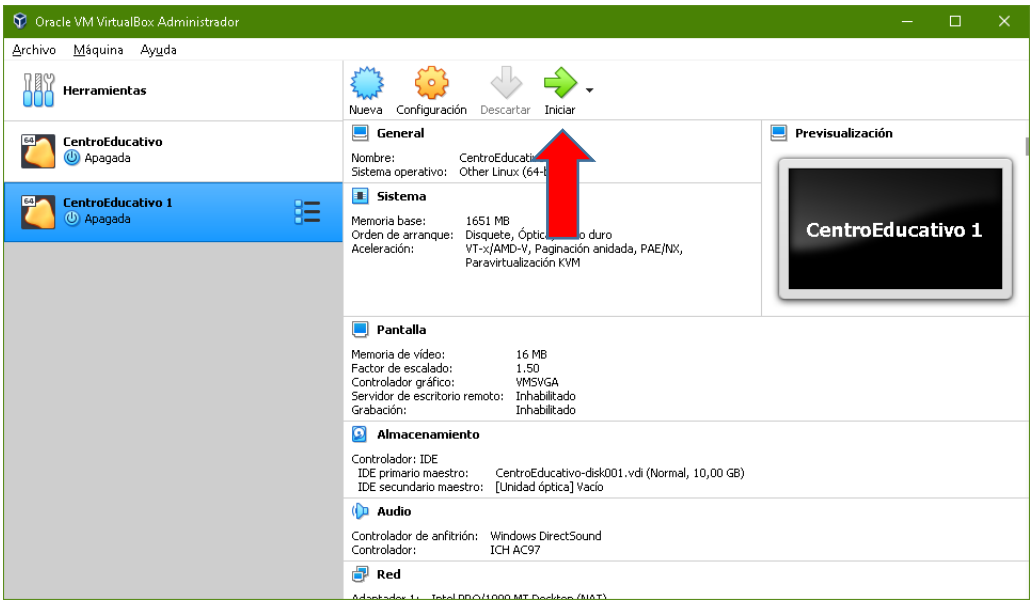

Figura A.1.10 Iniciar máquina virtual

9. Una vez cargada aparecerá la siguiente ventana y el usuario solo tendrá que abrir el navegador Mozilla Firefox, situado en el escritorio de la máquina virtual y pulsar el acceso directo llamado *Centro Educativo* para acceder a la herramienta y poder utilizarla.

Figura A.1.11 Abrir proyecto

10. Cuando el usuario ya no necesite seguir utilizando la aplicación, simplemente cierra la máquina virtual y pulsa la opción guardar estado de la máquina para que la próxima vez que la use no tarde tanto en iniciar.

A.2 Servidor local

Para instalar el programa que creará un servidor local para ejecutar el proyecto, es necesario descargárselo de la página https://www.apachefriends.org/es/index.html. Si el usuario tiene Windows como sistema operativo, deberá pulsar el enlace marcado con 1, si tiene Linux como sistema operativo, deberá pulsar el enlace marcado con 2.

Figura A.2.1 Enlaces de descarga

Una vez que se ha descargado el programa se procede a la instalación del mismo, para la cual el usuario debe hacer doble click en el archivo que se acaba de descargar y debe seguir los siguientes pasos.

1. En cuanto se ejecuta el archivo aparece la advertencia de Windows, el usuario debe pulsar *Sí* y esperar a que se abra el instalador. El usuario debe pulsar *Next* hasta que aparezca la siguiente pantalla.

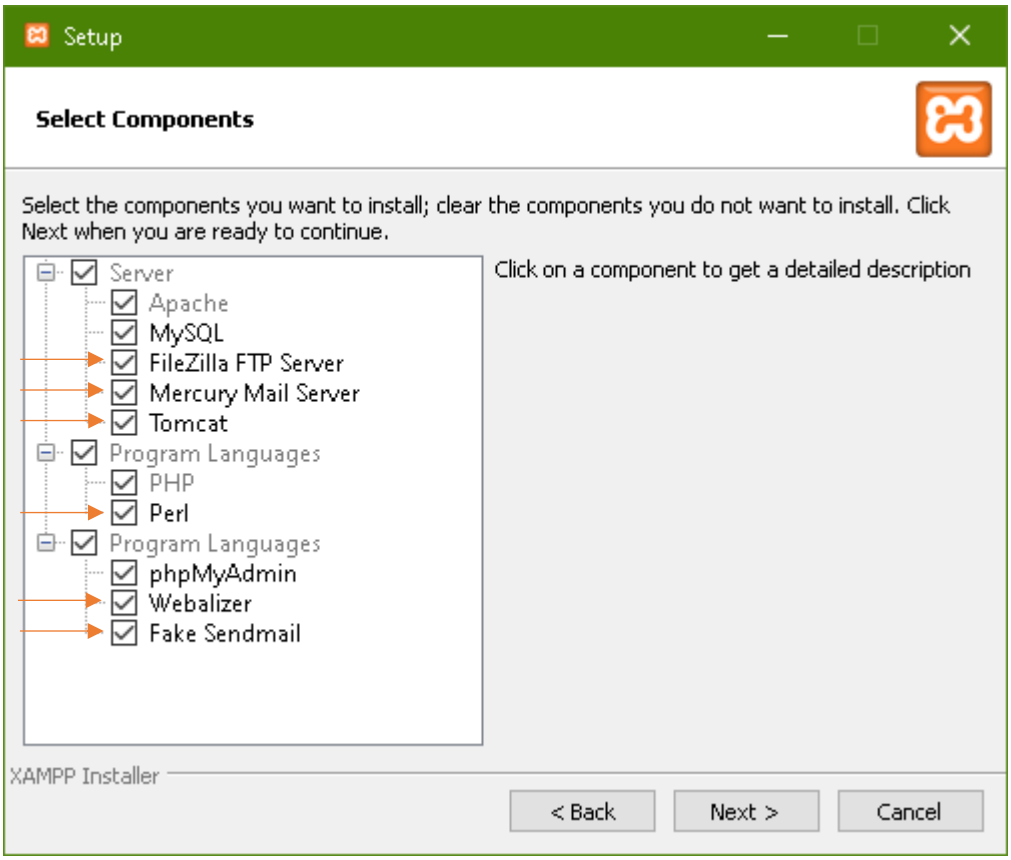

Figura A.2.2 Instalador

2. Aquí debe desmarcar las opciones llamadas *Filezilla FTP Server, Mercury Mail Server, Tomcat, Perl, Webalizer* y *Fake Sendmail,* ya que no son necesarias para ejecutar esta aplicación web. A continuación debe pulsar *Next* hasta que aparezca la siguiente pantalla, en la que debe desmarcar la opción marcada por la flecha y pulsar *Next* hasta que llegue a la página en la que se instala el programa.

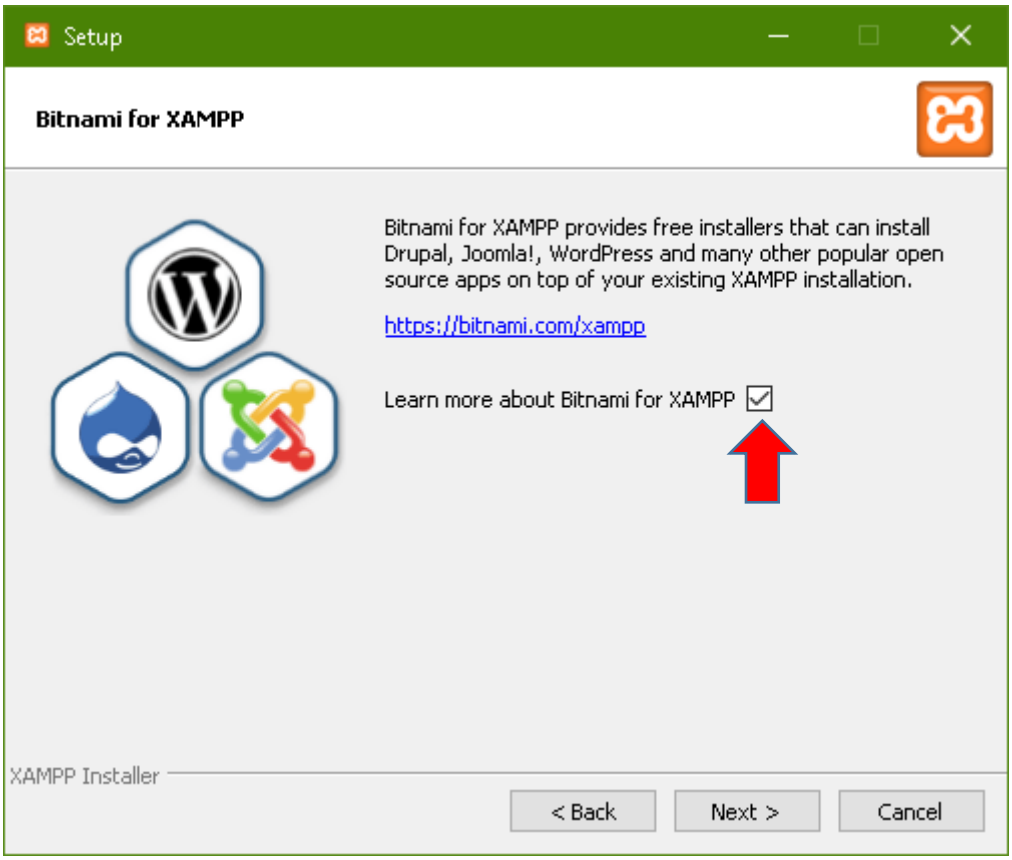

Figura A.2.3 Instalador (2)

3. El usuario debe esperar a que se instale el programa, el cual tardará varios minutos. Una vez instalado, aparecerá una pantalla y el usuario pulsará *Finish*. El programa se abrirá tras unos segundos de espera y aparecerá esta pantalla en su ordenador. El usuario pulsará el botón *Config*.

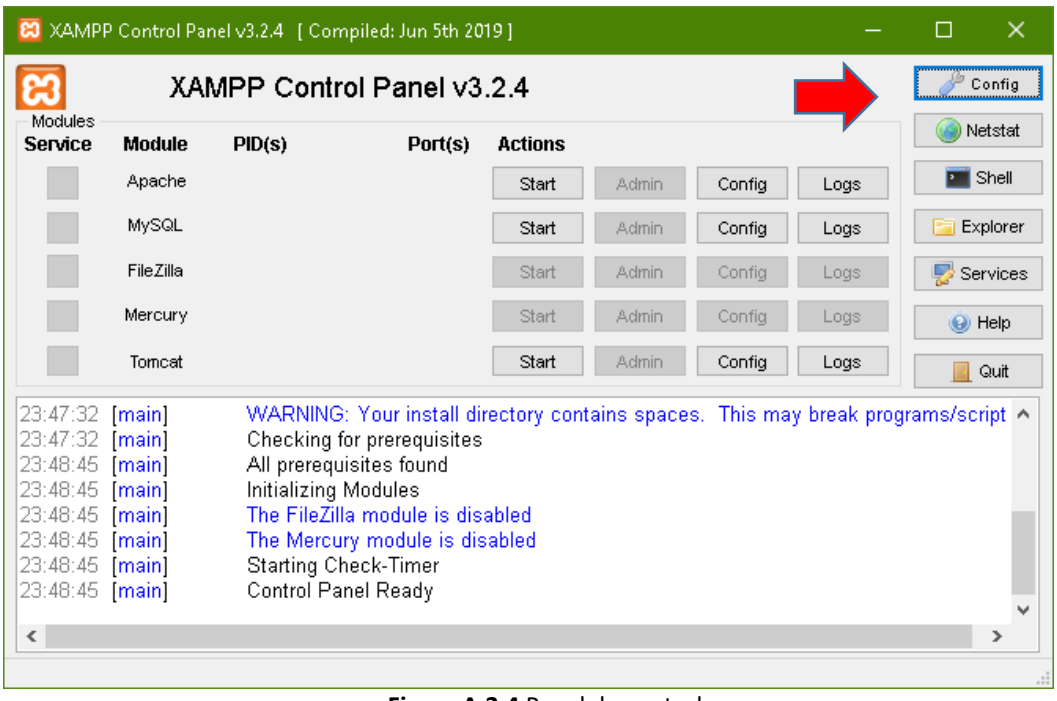

Figura A.2.4 Panel de control

4. Una vez se ha pulsado el botón aparecerá la siguiente pantalla de configuración en la que el usuario debe marcar las siguientes opciones y pulsar el botón *Save*.

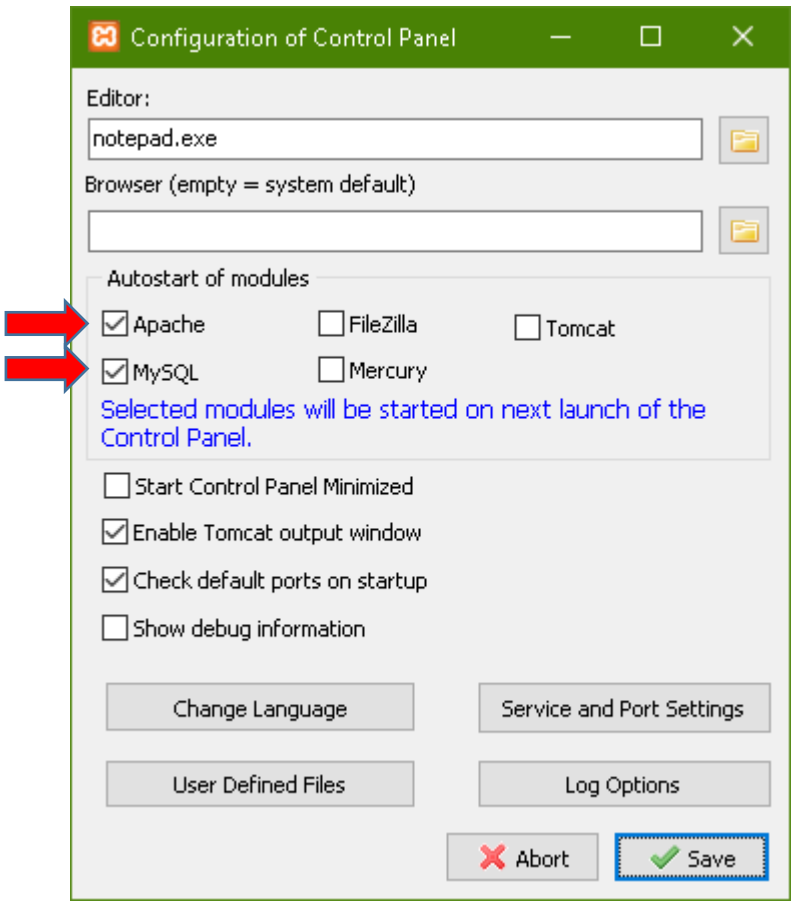

Figura A.2.5 Configuración

5. Si todo va bien, debería aparecer la siguiente alerta de Windows y el usuario debe pulsar *Permitir acceso.*

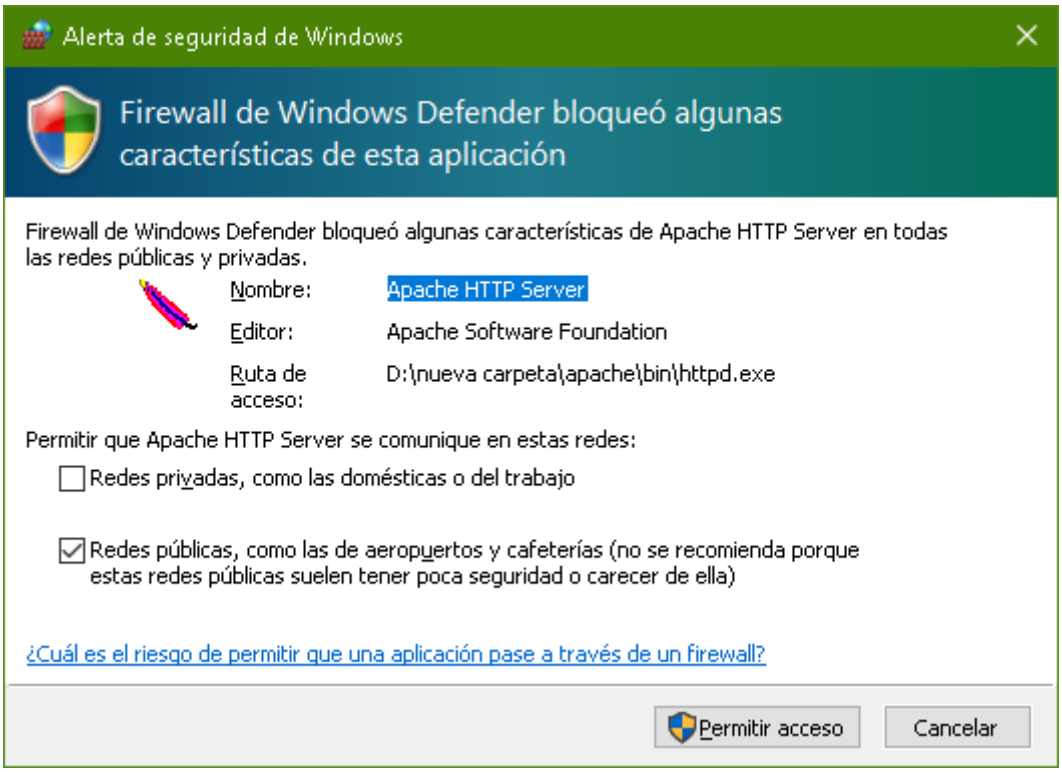

Figura A.2.6 Alerta firewall

6. Y la imagen final del panel de control con los servidores encendidos debería ser la siguiente. Si no es así, el usuario debe pulsar los botones *Start* correspondientes a la línea de *Apache* y *MySQL*.

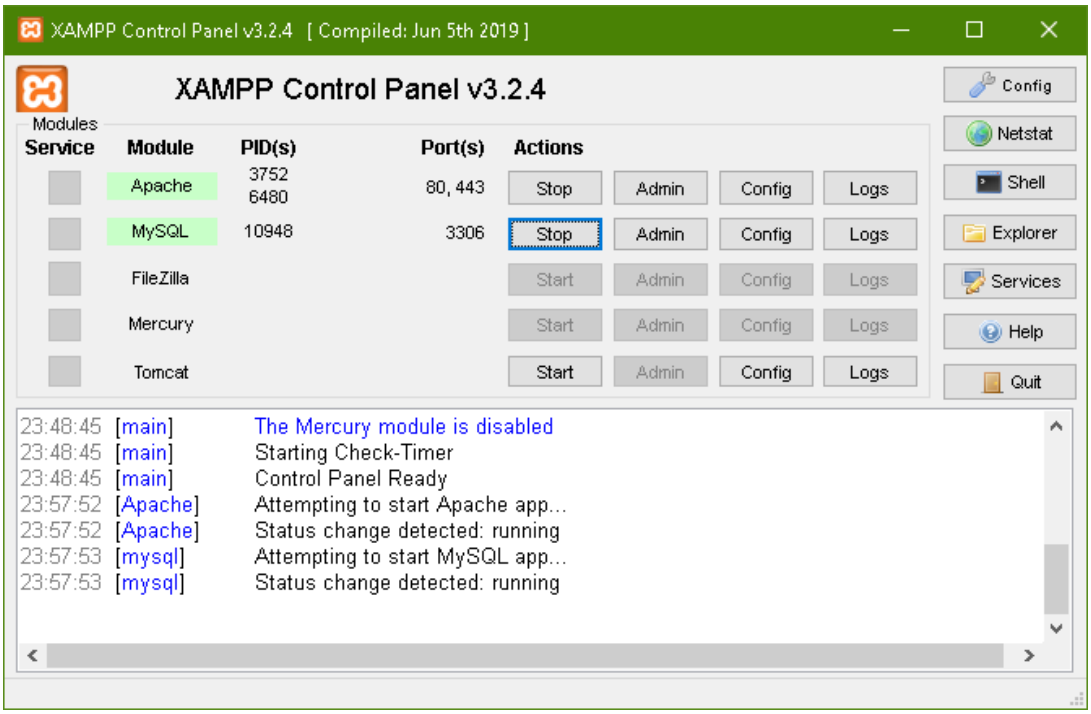

Figura A.2.7 Panel de control final

Para instalar la herramienta el usuario solo debe abrir el explorador de archivos e ir a la dirección C:\xampp\htdocs y copiar ahí la carpeta *CentroEducativo* que se extrajo del fichero zip.

Para comprobar que todo funciona correctamente, el usuario debe abrir un navegador web y poner en la barra de la URL la dirección http://localhost/CentroEducativo/ y pulsar la tecla *Intro*. Si todo funciona, el usuario debería visualizar la siguiente página que es el inicio de sesión de la herramienta.

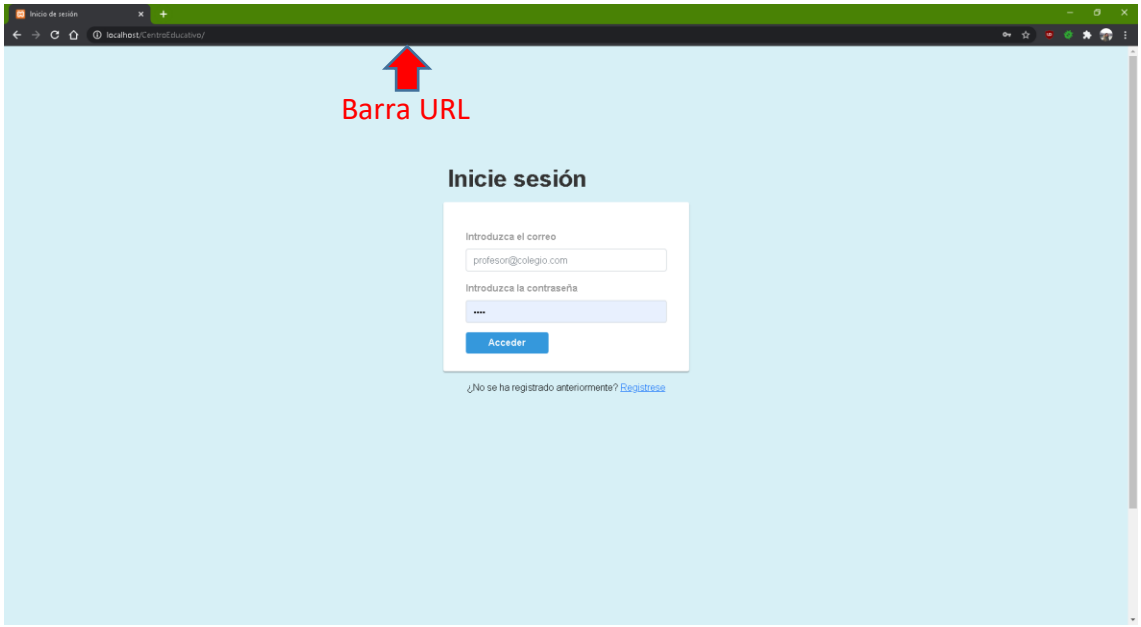

Figura A.2.8 Página inicio sesión

Apéndice B

Manual de usuario

En este apéndice se describirán los pasos para registrarse e iniciar sesión y, una vez que se accede a la página con el correo y la contraseña, para buscar, importar, añadir, editar, consultar o eliminar datos.

Ya que todos los formularios de los que se dispone en cada página de gestión son iguales, solo se detallarán las acciones para el apartado de gestión del alumnado que es el que posee todas las características.

Es importante que para que la aplicación funcione perfectamente y al usuario no le falten datos a la hora de añadir los alumnos y las asignaturas, los datos deben introducirse en el siguiente orden:

1. **Asignaturas**, ya sea mediante la importación o manualmente.
2. **Profesores**, deben añadirse manualmente ya que no van a cambiar mucho de un curso a otro.
3. **Unidades**, pueden añadirse mediante importación o manualmente.
4. **Alumnos**, mediante importación o manualmente.

B.1: Registro e inicio de sesión

La primera página que aparece en la aplicación es la de inicio de sesión. Si el usuario no se ha registrado, debe pulsar el enlace que indica la flecha.

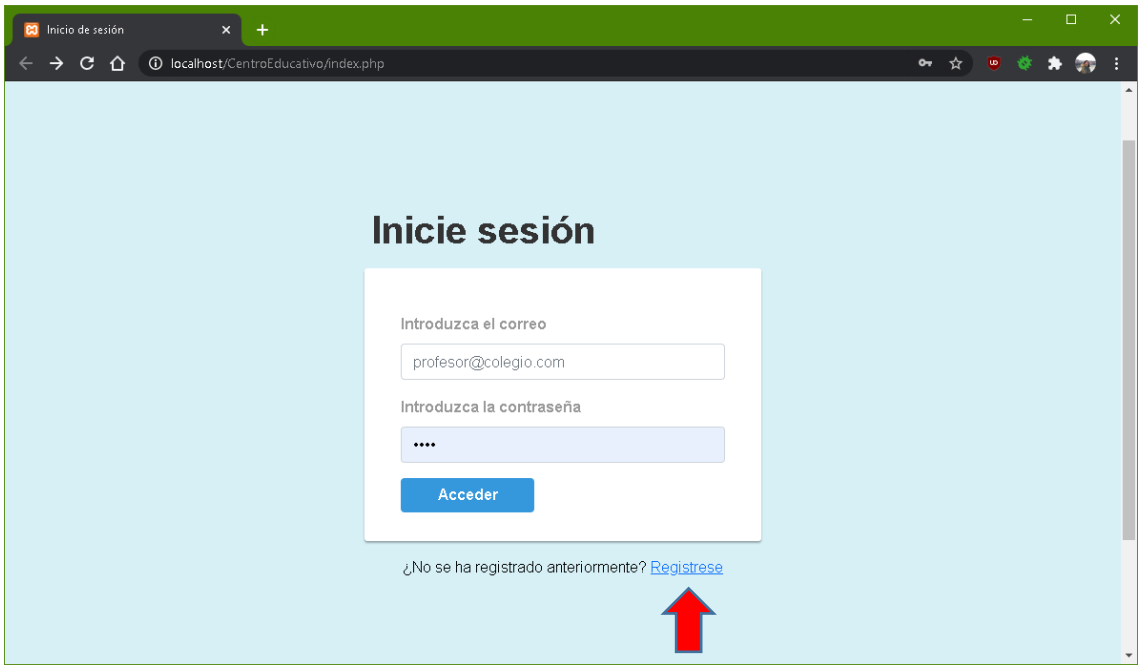

Figura B.1 Registro de usuario (1)

Al pulsar el enlace, aparece el siguiente formulario para introducir el nombre de usuario, correo y contraseña y se pulsa el botón Registrarse. Una vez que el usuario se ha registrado, ya accede a la página principal.

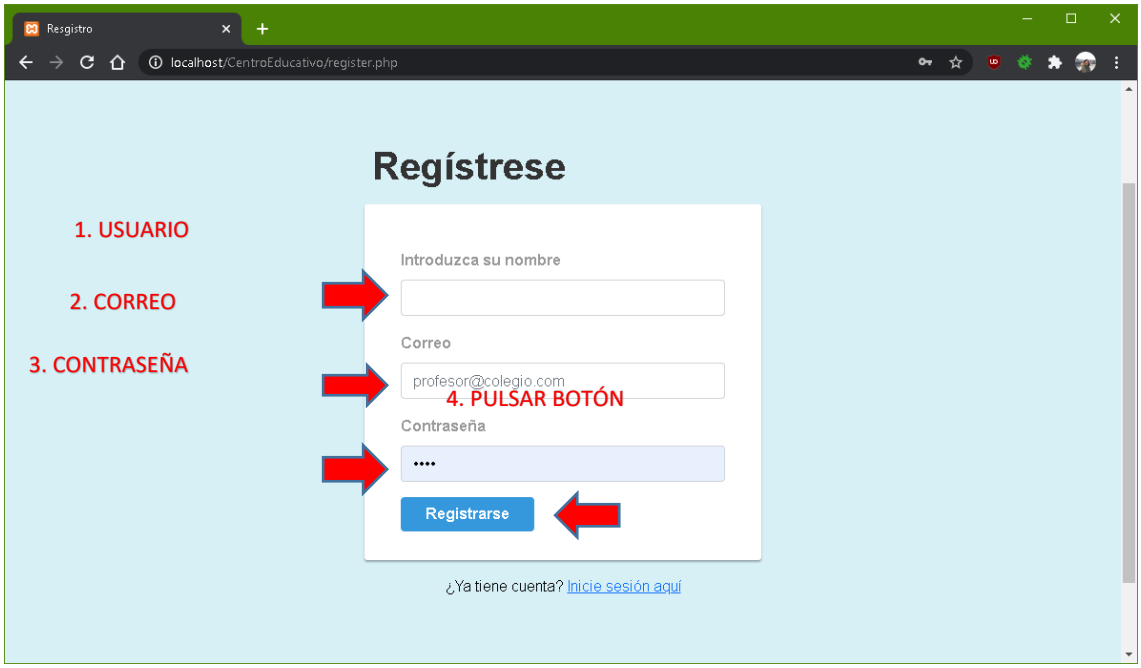

Figura B.2 Registro de usuario (2)

Si el usuario ya se ha registrado previamente, solo tiene que rellenar que la siguiente página, incluyendo el correo, la contraseña y pulsar el botón Acceder.

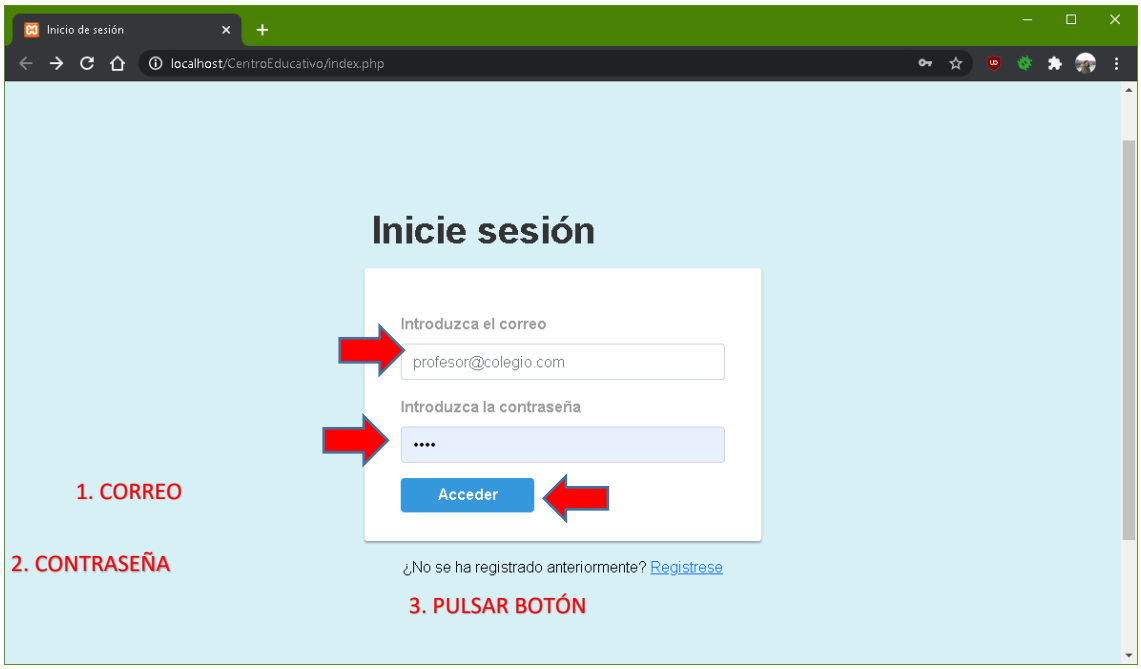

Figura B.3 Inicio de sesión

B.2: Página principal

Para acceder a una sección específica, el usuario solo debe pulsar el botón de la sección correspondiente. Si desea cerrar sesión, debe pulsar el botón indicado por la flecha.

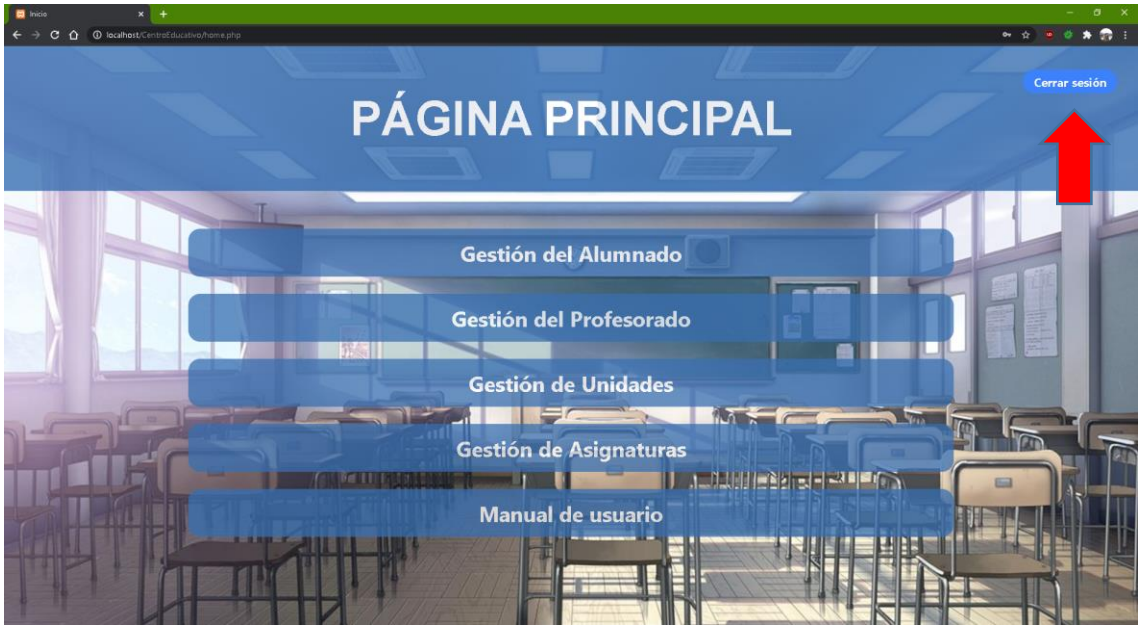

Figura B.4 Página principal

B.3: Buscador

Si el usuario desea buscar cualquier dato en la tabla, debe introducir lo que quiere buscar en el lugar señalado por la flecha. No es necesario pulsar ningún botón, en cuanto vaya escribiendo el buscador se irá actualizando. Si desea buscar un nombre completo, debe escribirlo en el mismo formato en el que está el nombre de la imagen del ejemplo.

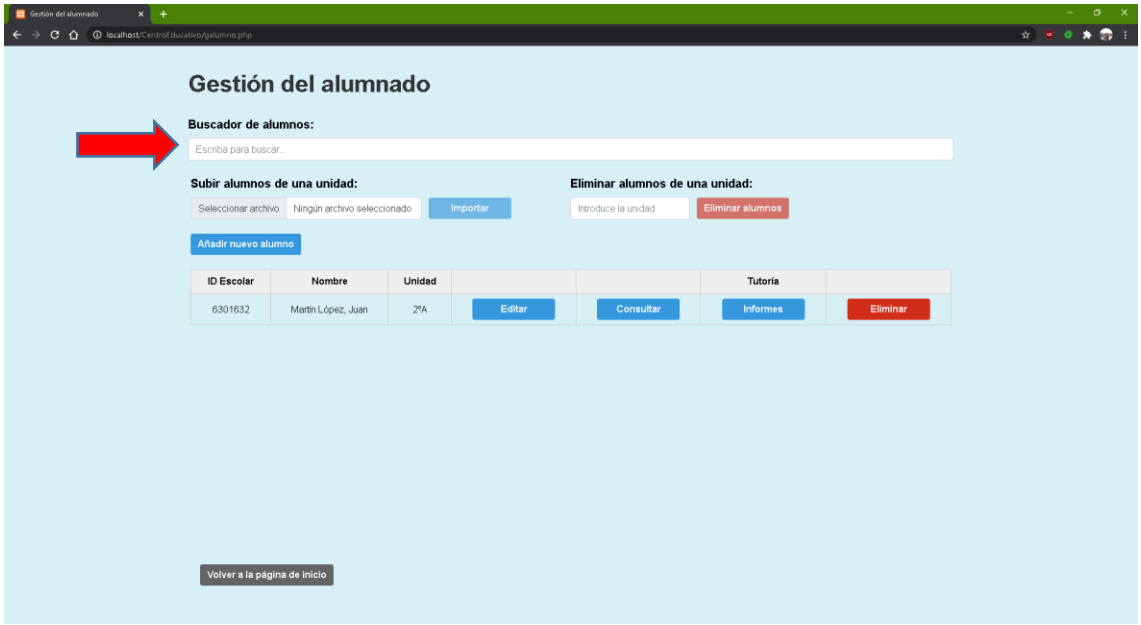

Figura B.5 Gestión alumnado buscador

B.4: Importador

Si el usuario desea importar un archivo, solo debe pulsar el botón seleccionar archivo (marcado con el número 1). A continuación se abrirá el explorador de archivos y el usuario seleccionará el archivo que desea subir y pulsará abrir. Tras esto, el usuario pulsará el botón Importar (marcado con el número 2) y el archivo se importará.

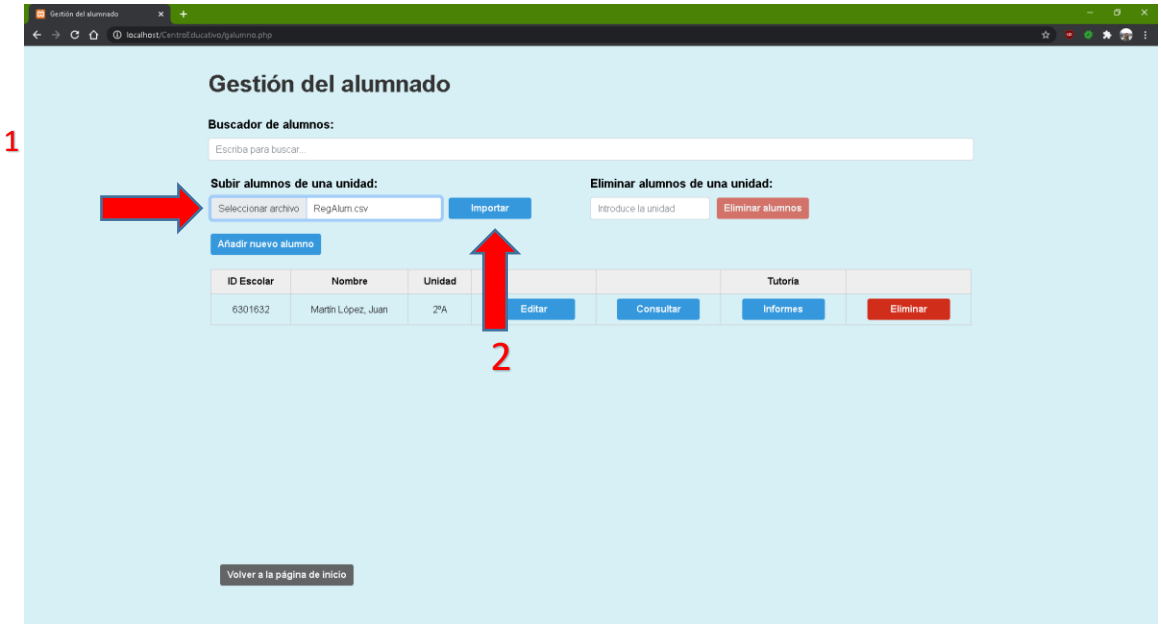

Figura B.6 Gestión alumnado importador (1)

B.5: Eliminar alumnos de una unidad

Esta funcionalidad es propia del apartado de Gestión del alumnado. Si el usuario desea eliminar los alumnos de una unidad solo debe introducir la unidad en el cuadro de texto marcado con 1 y pulsar el botón Eliminar alumnos marcado con 2.

Figura B.7 Gestión alumnado eliminar alumnos unidad

B.6: Añadir

Si el usuario desea añadir un nuevo alumno manualmente, solo debe pulsar el botón Añadir nuevo alumno (marcado con 1) que le llevará a una página donde se mostrará los campos con los datos del alumno para rellenar.

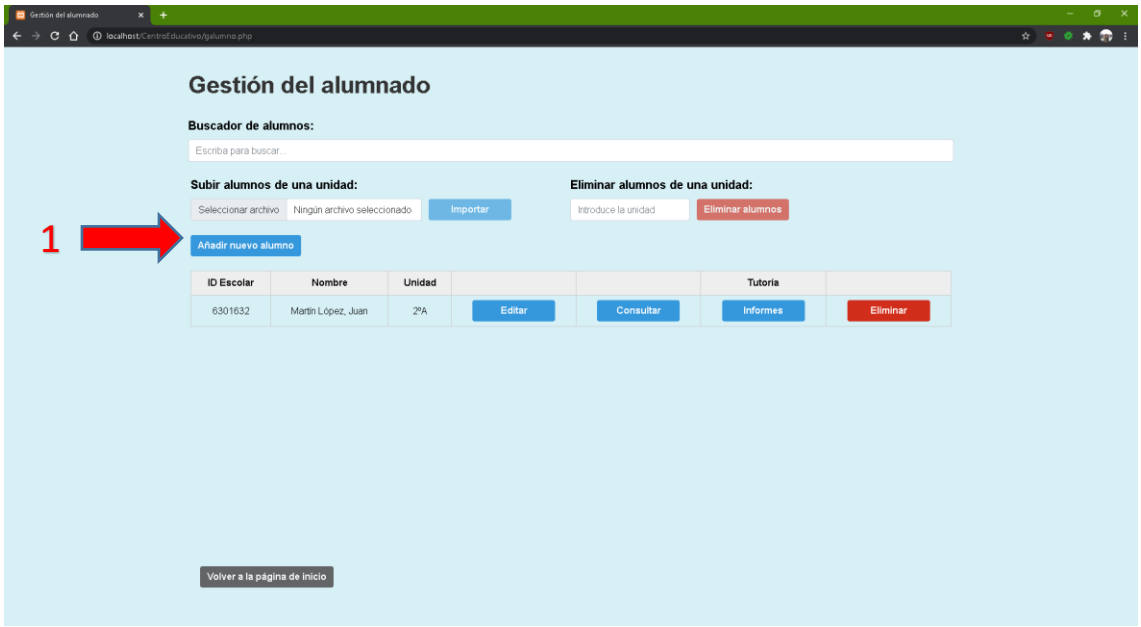

Figura B.8 Gestión alumnado añadir alumno (1)

Una vez se hayan rellenado todos los datos en el formulario, el usuario pulsará el botón Añadir (marcado con un 2), que guardará los datos y llevará al usuario a la página de gestión en la que se encuentre.

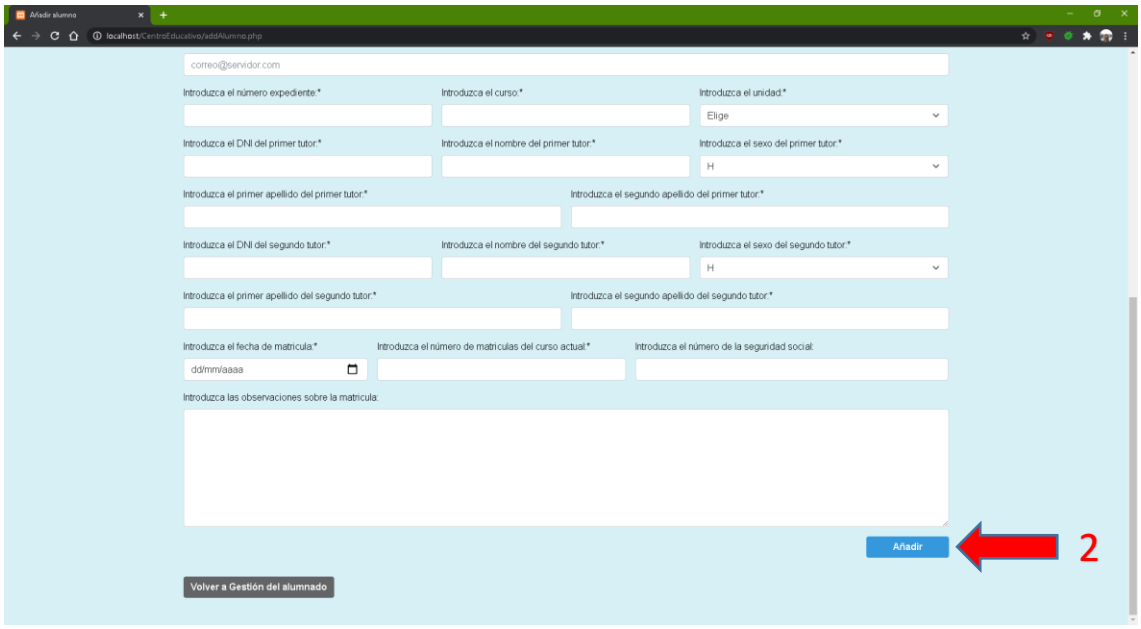

Figura B.9 Gestión alumnado añadir alumno (2)

B.7: Editar

Si el usuario desea editar los datos de un alumno, solo debe pulsar el botón Editar (marcado con 1) que le llevará a una página donde se mostrará los campos con los datos del alumno para rellenar.

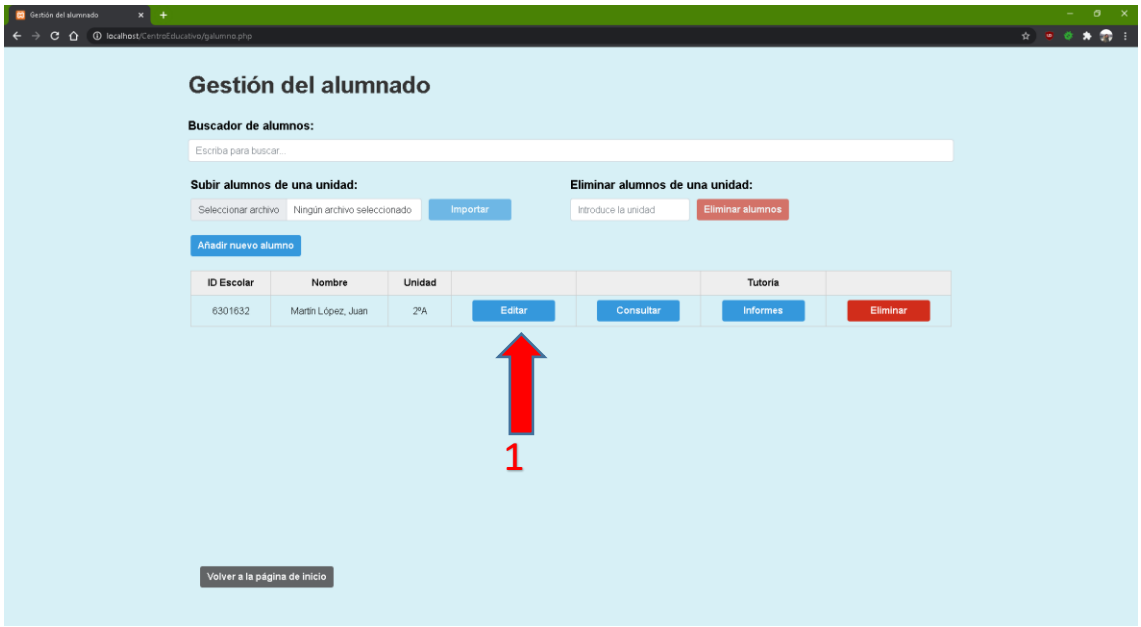

Figura B.10 Gestión alumnado editar alumno (1)

Una vez se editado alguno de los datos del formulario, el botón Editar (marcado con un 2), se habilitará, el usuario lo pulsará, se guardarán los datos y llevará al usuario a la página de gestión en la que se encuentre.

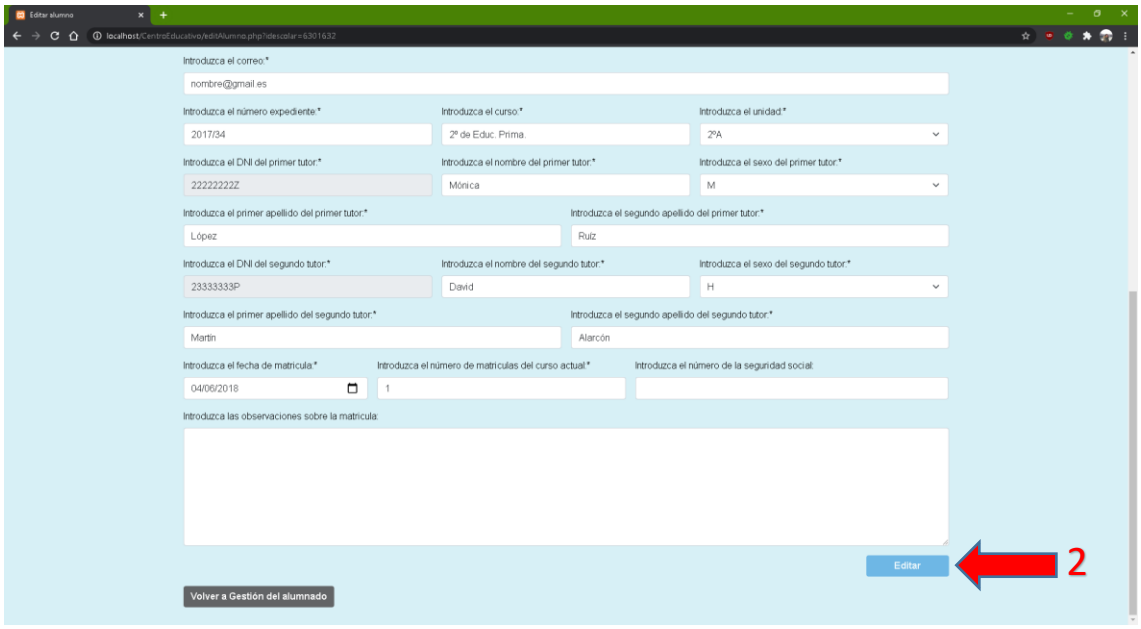

Figura B.11 Gestión alumnado editar alumno (2)

B.8: Consultar

Si el usuario desea consultar los datos del alumno, solo debe pulsar el botón Consultar (marcado con la flecha) que le llevará a una página parecida a la de Editar, donde se mostrará los datos del alumno.

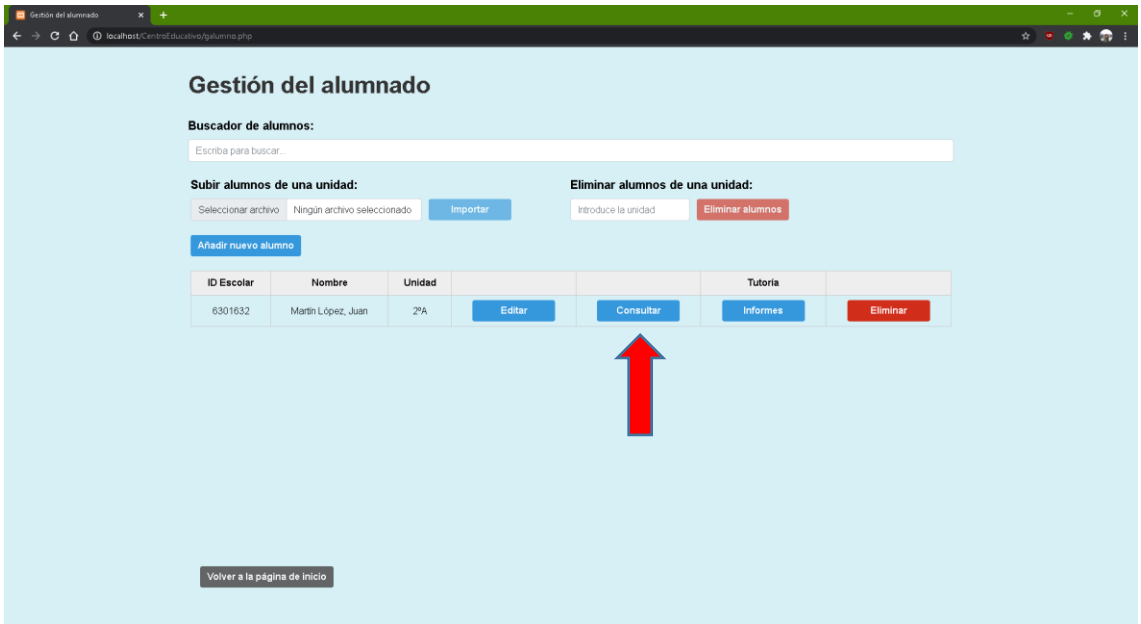

Figura B.12 Gestión alumnado consultar datos de alumno

B.9: Informes de tutoría

Esta funcionalidad es propia del apartado de Gestión del alumnado. Si el usuario desea consultar los informes del alumno, solo debe pulsar el botón Informes (marcado con la flecha) que le llevará a una página parecida a esta donde se encuentran los informes del alumno.

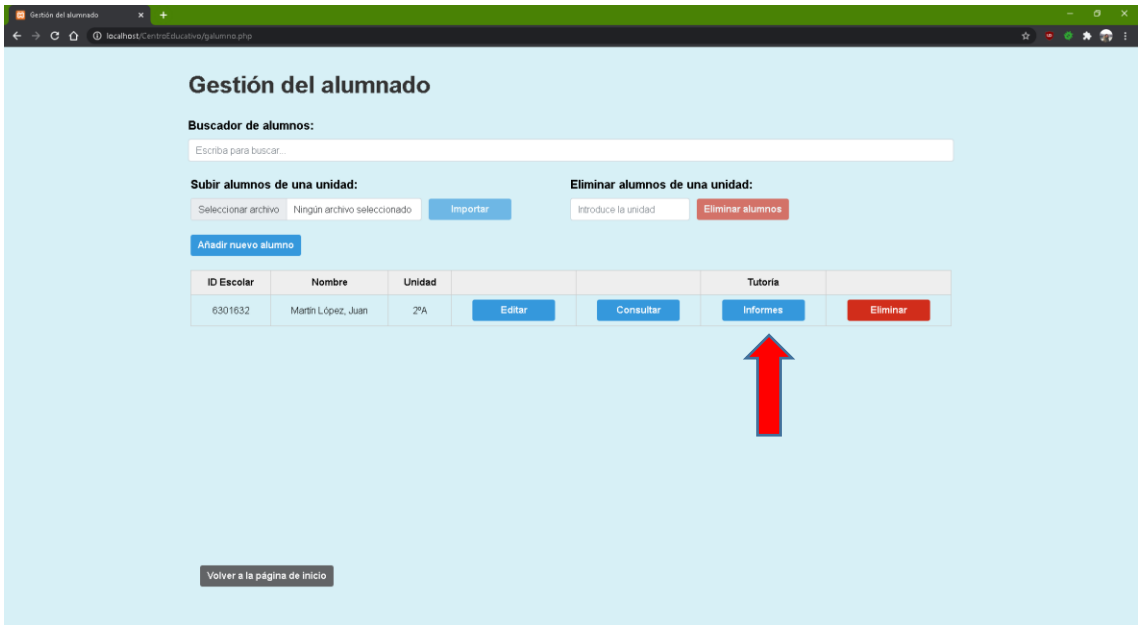

Figura B.13 Gestión alumnado informes tutoría de alumno

B.10: Eliminar

Si el usuario desea eliminar el alumno solo debe pulsar el botón Eliminar (marcado con la flecha) y aparecerá un mensaje preguntando si desea eliminar. Si el usuario pulsa Ok, el alumno se eliminará, si pulsa Cancel o lo cierra, el usuario no se eliminará.

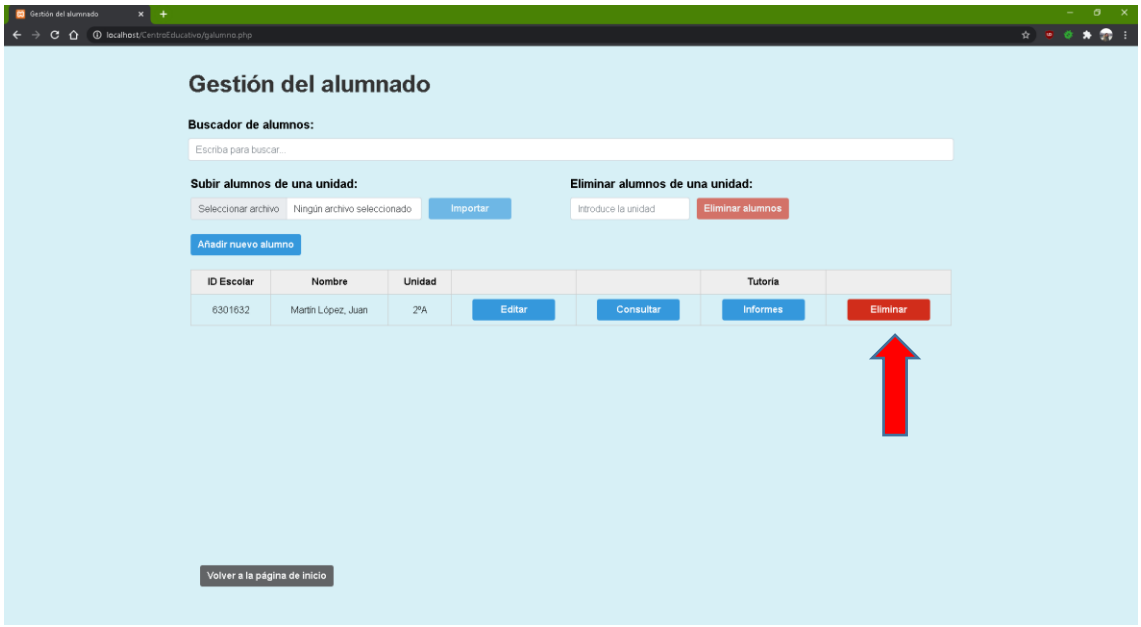

Figura B.14 Gestión alumnado eliminar alumno

B.11: Marcar profesor como ausente

El usuario busca el profesor que desea marcar como ausente y pulsa el botón *Editar*. Una vez aparece el formulario, el usuario busca la sección *Estado del profesor* y despliega la lista para seleccionar *Ausente.*

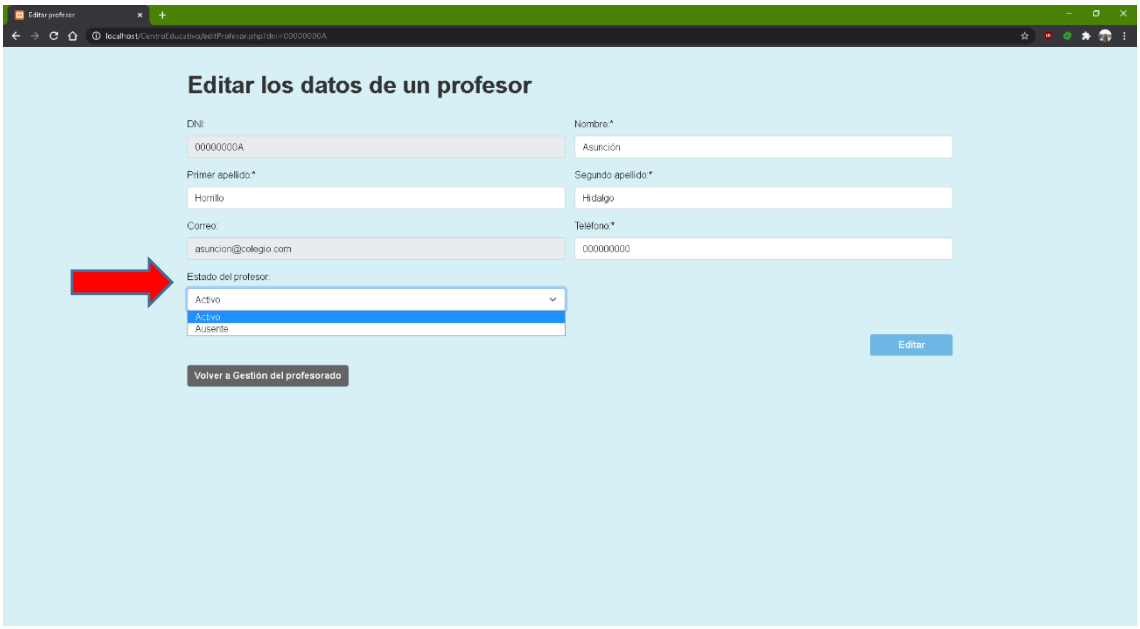

Figura B.15 Profesor estado activo

Una vez se ha marcado el profesor como ausente, aparecerá un cuadro de texto para que el usuario lo rellene con la información que necesita el profesor que lo supla durante su ausencia. Una vez lo ha rellenado pulsa el botón *Editar* y el sistema guarda los datos.

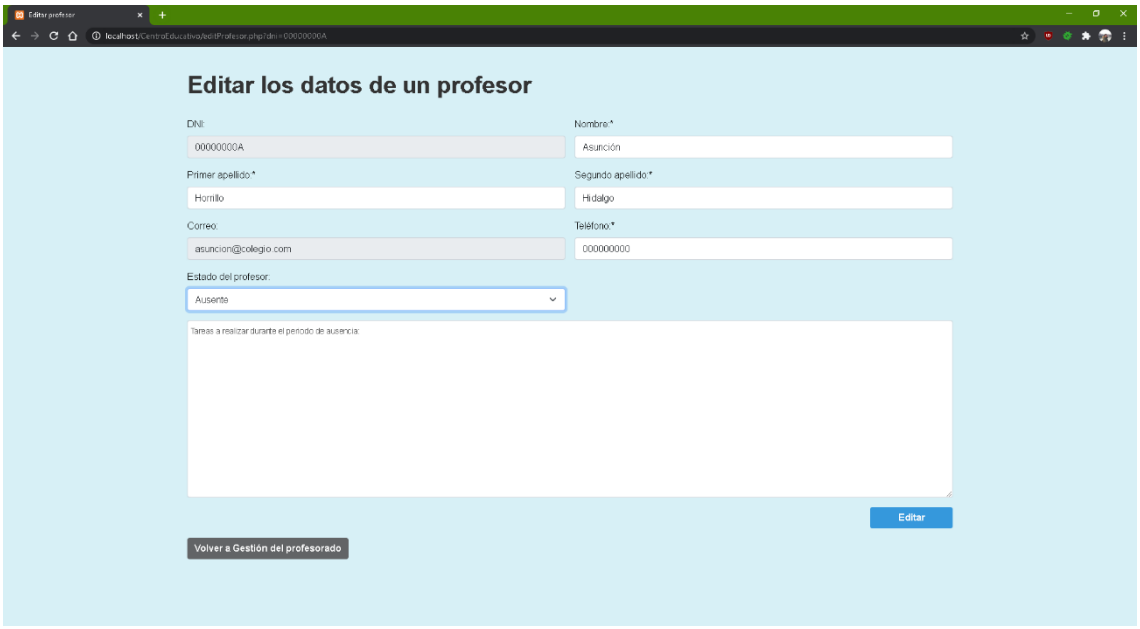

Figura B.16 Profesor estado ausente